Doctoral Thesis

A Study on Private and Secure Federated Learning

Fumiyuki Kato

February 2024

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Doctoral Thesis
submitted to Department of Social Informatics,
Graduate School of Informatics,
Kyoto University
in partial fulfillment of the requirements for the degree of
DOCTOR of INFORMATICS

Thesis Committee: Takayuki Ito, Professor
Tomohiro Kuroda, Professor
Yasuo Okabe, Professor
Masatoshi Yoshikawa, Professor Emeritus

A Study on Private and Secure Federated Learning*

Fumiyuki Kato

Abstract

Federated Learning (FL) has emerged as a promising new machine learning paradigm for collaborative learning with privacy considerations among different parties such as different institutions, devices, etc. The basic scheme includes a single central server that orchestrates the whole training process and many clients who have their personal or private data as training data. We cooperatively train a single global model according to distributed optimization algorithms without directly sharing the training data. In FL, only gradient information used in the optimization process is shared in place of raw training data itself, thereby freeing the party training the model from the cost of managing private data. With the growing concerns about privacy regulations for large-scale data analysis, as exemplified by GDPR, FL has attracted significant attention from both academia and industry.

However, FL has various privacy risks due to its distributed architecture, even though it is supposed to be a privacy-aware machine learning scheme. Specifically, there is no strict privacy guarantee for the trained model itself even if we use FL, and the trained model once released can leak sensitive information about the training data. Additionally, private information can be leaked from the gradient information exchanged between the distributed parties in the training phase. Furthermore, in addition to the privacy leaks, there are security risks. Due to the distributed nature of the scheme, the clients may not trust the server and the clients also could be untrusted. They may deviate from established protocols and improperly control the behavior of the model. In other words, FL is an inadequate scheme in terms of privacy and security.

*Doctoral Thesis, Department of Social Informatics, Graduate School of Informatics, Kyoto University, KU-I-DT6930-33-0175, February 2024.

Therefore, in this thesis, towards realizing the private and secure FL, we aim to comprehensively overcome these imperfections by examining advanced privacy-preservation techniques, with a particular focus on Differential Privacy (DP) and Trusted Execution Environment (TEE). The goal of this thesis is to answer the following main research question: *To enhance privacy and security in Federated Learning, how can advanced privacy-preservation techniques such as Differential Privacy and Trusted Execution Environment be effectively integrated into Federated Learning?* To answer this question, we pick out the key privacy/security properties in FL from existing studies and from use cases and clarify what needs to be done. Then, we design the following three novel FL/FA frameworks ULDP-FL (Chapter 3), OLIVE (Chapter 4) and VLDP (Chapter 5), utilizing DP and/or TEE, or alternative MPC techniques, to overcome the weakness of security and privacy aspects of FL.

In Chapter 3, we examine rigorous privacy protection for models trained in FL with DP. In particular, we target the general setting of cross-silo FL, where each participating client corresponds to an institution of a certain size, with user-level DP guarantees. User-level DP is a practical definition of DP, which guarantees indistinguishability for all records held by the user instead of a single record in the original DP. Under this setting, we show that existing algorithms can only achieve impractical privacy guarantees and propose algorithms that offer better privacy-utility tradeoffs. Our proposed method directly guarantees user-level DP by applying per-user weighted clipping to the existing de facto DP-FedAVG. Furthermore, we propose a utility-boosting weighting method and develop an MPC protocol to achieve it under a more stringent trust model.

In Chapter 4, we focus on the server-side TEE in FL, which enables the guaranteeing of the privacy of the shared gradients to the central untrusted server and to provide better utility of differentially private FL. While TEE provides another level of security for FL, TEE itself is known to have a fundamental vulnerability: memory access leaks. Through analysis of memory access patterns for FL aggregation operations, we discover the possibility of privacy risks when sparsified gradients are used. Using the observable memory access pattern information, we design a novel attack that reveals private data and show the effectiveness through experiments using real-world data. To defend against this attack, we design an oblivious algorithm such that the memory access patterns resulting from FL aggregation operations are independent of the input data. Finally, we evaluate the

proposed oblivious algorithm on a real data scale and show its efficiency.

In Chapter 5, we address the defense against malicious clients who deviate from the supposed protocol to control the behavior of the output of federated analytics tasks. In particular, to begin the initial research on this type of attack, we focus on a simple Federated Analytics task, i.e., a frequency estimation under local differential privacy (LDP). Because LDP requires perturbation of data on the client side, the central server does not have complete control over this protocol, allowing a malicious client to control the final estimates on server side. We show that the attack can be partially prevented by developing a verifiable LDP protocol that targets simple LDP protocols. We believe this type of client-side verifiability that we propose in this method can be extended to prevent attacks in more complicated federated tasks including FL in the future.

Finally, in Chapter 6, we discuss the profound social impact of our comprehensive studies produced in this thesis. Furthermore, we argue the specific benefits on the some fields where the real-world FL applications have been studied. We believe the research presented in this thesis will make a significant contribution to operationalizing privacy-preserving FL as a production-level application. As an overall conclusion, we present state-of-the-art methodologies that combine advanced privacy-preserving technologies to address the privacy and security challenges in FL, by resolving fundamental issues such as lack of strict privacy guarantees, protection against vulnerabilities, and the hardness of establishing a reliable defense mechanism.

Additionally, in Appendices A and B, we present our independent studies on DP and TEE, respectively. We consider these two techniques to be important because they are the core of this thesis from a technical perspective.

Keywords: Privacy-preserving Machine Learining, Federated Learning, Federated Analytics, Differential Privacy, Trusted Execution Environment

Contents

1 Introduction **1**

1.1 Background 1

1.1.1 Privacy and Security problems in FL 3

1.2 Differential Privacy 5

1.3 Trusted Execution Environment 6

1.4 Integration into FL 7

1.5 Contributions 12

1.5.1 Across-silo User-level DP in cross-silo FL 12

1.5.2 FL on TEE against the risk of sparsification 13

1.5.3 Verifiable LDP protocol against Untrusted Client 13

2 Literature Review **15**

2.1 Federated Learning 15

2.2 Differential Privacy 18

2.2.1 Differentially Private Federated Learning 22

2.2.2 Differentially Private Federated Analytics 24

2.3 Trusted Execution Environment 25

2.3.1 Federated Learning with TEE 26

3 ULDP-FL: Federated Learning with Across Silo User-Level Differential Privacy **28**

3.1 Introduction 28

3.2 Background & Preliminaries 31

3.2.1 Cross-silo Federated learning 31

3.2.2 Differential Privacy for Multiple Records 31

3.2.3 DP-FL in Cross-silo Setting 34
3.3 ULDP-FL Framework 34
3.3.1 Trust Model and Assumptions 34
3.3.2 Privacy Definition 35
3.3.3 Baseline Methods: ULDP-NAIVE/GROUP-k 36
3.3.4 Advanced methods: ULDP-AVG/SGD 39
3.4 Private weighting protocol 51
3.4.1 Correctness 55
3.4.2 Privacy 57
3.5 Experiments 57
3.5.1 Settings 57
3.5.2 Results 60
3.6 Conclusion 68
3.6.1 Future works 68

4 OLIVE: Oblivious Federated Learning on Trusted Execution Environment against the risk of sparsification 69
4.1 Introduction 69
4.2 Preliminaries 73
4.2.1 Federated Learning with Sparsification 73
4.2.2 Memory Access Pattern Leakage of TEE 73
4.3 Proposed System 76
4.3.1 Scenario 76
4.3.2 System Overview 77
4.3.3 Security Analysis 80
4.4 Attack on Gradient index 83
4.4.1 Design 83
4.4.2 Evaluation Task 87
4.4.3 Empirical Analysis 87
4.5 Oblivious Algorithms 91
4.5.1 Baseline method 92
4.5.2 Advanced method 92
4.5.3 Optimization 96
4.5.4 Relaxation of Obliviousness 97
4.5.5 Experimental results 98

4.5.6 Discussion . . . 101
4.6 Relationship with DP-FL. . . . 102
4.6.1 Attack evaluation . . . 105
4.7 Related works . . . 107
4.8 Conclusion . . . 109
4.8.1 Future works . . . 110
4.9 Appendix . . . 111
4.9.1 Model architectures . . . 111

5 VLDP: Preventing Manipulation Attack in Local Differential Privacy Using Verifiable Randomization Mechanism 113
5.1 Introduction . . . 114
5.2 Attacks on LDP protocols . . . 116
5.2.1 Local Differential Privacy . . . 116
5.2.2 Attacks on LDP protocols . . . 119
5.3 Problem Statements . . . 121
5.3.1 Overview of the goal . . . 121
5.3.2 Security definitions . . . 123
5.4 Proposed Method . . . 125
5.4.1 Secure kRR . . . 125
5.4.2 Secure OUE . . . 129
5.4.3 Secure OLH . . . 131
5.5 Evaluation . . . 132
5.6 Conclusion . . . 134
5.6.1 Future works . . . 135

6 Discussion 136
6.1 Social Impact . . . 136
6.2 Real-World FL Application . . . 139
6.2.1 Application for Mobile Devices . . . 139
6.2.2 Application in Industrial Engineering . . . 140
6.2.3 Application in Healthcare . . . 141
6.3 Limitation . . . 142

7 Concluding Remarks 143

Acknowledgements **145**

References **147**

List of Publications **188**

Appendix **190**

A HDPView: differentially private materialized view for exploring high dimensional relational data 190

A.1 Introduction 190

A.2 Related Works 195

A.3 Preliminaries 197

A.4 Problem Formulation 198

A.5 Proposed Algorithm 200

A.6 Evaluation 211

A.7 Conclusion 222

B PCT-TEE: Trajectory-based Private Contact Tracing System with Trusted Execution Environment 222

B.1 Introduction 222

B.2 Preliminaries 227

B.3 Problem Formulation 230

B.4 System Overview 236

B.5 Spatiotemporal PSI 239

B.6 Experiments 252

B.7 Related Works 265

B.8 Conclusions 268

List of Figures

1.1 Overview of Federated Learning (FL) architecture. A central server (top) that orchestrates the whole training process and many clients (bottom) who have their personal or private data as training data collaboratively train a single global model with local model update for each client. The red balloons indicate the privacy and security risks in FL. 2

3.1 In cross-silo FL, in general, records belonging to the same user can exist across silos, e.g., a user can use several credit card companies. In this study, we investigate how to train models satisfying *user-level* DP in this setting. 29

3.2 Group-privacy conversion results. 33

3.3 An intuitive illustration of the difference between ULDP-NAIVE and ULDP-AVG. In ULDP-NAIVE, every single user can contribute to all model deltas. On the other hand, in ULDP-AVG, one user's contribution is limited to a small portion, i.e., $1/|U|$ of the whole model delta. This reduces sensitivity. When $|U|$ is large, which often happens in cross-silo FL, it can be a particular advantage. 43

3.4 Record distribution across users and silos. 59

3.5 Example of record allocation on Creditcard. 59

3.6 Privacy-utility trade-offs on Creditcard dataset: Accuracy (Left), Privacy (Right). 61

3.7 Privacy-utility trade-offs on MNIST dataset: Test Loss (Left), Accuracy (Middle), Privacy (Right). 61

3.8 HeartDisease. 62
3.9 TcgaBrca. 62
3.10 Test loss of Creditcard: Weighting method is effective, especially in skewed distribution in many silos. 64
3.11 Test loss of TcgaBrca: the better weighting is effective especially in biased distribution. 64
3.12 In Creditcard, user-level sub-sampling achieves a more competitive privacy-utility trade-off. 65
3.13 In MNIST, user-level sub-sampling achieves a more competitive privacy-utility trade-off. 65
3.14 The dominant execution time of training grows linearly with parameter size and/or the number of users. 66
3.15 With a small model, the private weighting protocol has a practical execution time. 67

4.1 OLIVE, i.e., **O**b**LIV**ious f**E**derated learning on TEE is the first method of its kind to prevent privacy risks caused by the leakage of memory access patterns during aggregation in FL rigorously. This allows, for example, to enjoy utility of CDP-FL without requiring a trusted server like LDP-FL. 70
4.2 Dense gradients induce uniform access patterns. 80
4.3 Sparse gradients induce biased access patterns. 80
4.4 Overview of the proposed attack: Attacker is mainly assumed to access global test dataset, and aggregated model of each round, and the attacker trains classifier for each round (or all in batch) that takes parameter indices as input and outputs the target label set, observes memory access pattern through side-channels and extract parameter indices for each target participant and infers target label set. 85
4.5 Attack results on datasets with a fixed number of labels: Vulnerable, especially when there are few labels. 88
4.6 Attack results on datasets with a random number of labels (more difficult setting): When the number of labels is low, the attacker can attack the client without knowing the exact number of labels. 89

4.7 Attack results w.r.t. sparse ratios: Higher the sparsity, the more successful the attack tends to be. 89
4.8 Cacheline-level leakage on CNN of CIFAR10: Attacks are possible with at least slightly less accuracy. 90
4.9 The size of data that an attacker needs to access to achieve high success rate can be very small. 91
4.10 Running example of Advanced (Algorithm 13) at $n = 3$ (#user), $k = 2$ (#sparsified dimension), $d = 4$ (#dimension). 95
4.11 Performance results on a synthetic dataset w.r.t. models of various sizes: *Advanced* functions efficiently. α (sparse ratio) $= 0.01$ and n (number of clients per round) $= 100$. 99
4.12 Performance results w.r.t. various numbers of clients (N) at low sparsity ($\alpha = 0.1$): the *Advanced* gradually worsens with increasing number of clients. 100
4.13 The effects of optimizing the *Advanced* on MLP models on MNIST (left) and CIFAR100 (right). 100
4.14 Attack results on datasets with a fixed number of labels with DP ($\sigma = 1.12$). 106
4.15 Attack results on datasets with a random number of labels (more difficult setting) with DP ($\sigma = 1.12$). 106
4.16 Attack performance with variable noise multiplier σ. At realistic noise scales, the attack performance remains high. 107
4.17 Effective noise scales in defensing do not provide sufficient utility. 107
4.18 Test losses for each noise multiplier σ. 108

5.1 From top to bottom, normal protocol, input-manipulation attack and output-manipulation attack against an LDP protocol. 122
5.2 In secure kRR, with a sufficiently large *width*, categorical distribution by Algorithm 15 can accurately approximate the LDP distributions (left and middle). In secure OUE, it is almost exact discrete approximation with relatively small *width*. (right) 133
5.3 With the same *width*, the communication costs of kRR and OLH are small. However, OUE can approximate LDP accurately with small widths (Figure 5.2). 134

5.4 The characteristics of runtime is similar to bandwidth. OLH takes a little longer because of the hashing. 134

A.1 Data exploration through a *privacy-preserving materialized view* (*p-view* for short) of a multidimensional relational data. The p-view works as an independent query system. Analysts can explore sensitive and multidimensional data by issuing any range counting queries over the p-view before downstream data science workflows. 191
A.2 HDPView efficiently discovers blocks (i.e., groups of count cells) with smaller AEs (black arrow) and averages over each block with injected noise (red arrow). The p-view stores the randomized counts in a blockwise way. 200
A.3 Relative RMSEs against HDPView on all the datasets and workloads: HDPView shows small errors for a wide variety of high-dimensional range counting queries. 214
A.4 Examples of HDPView (left) and Privtree (right) on 2D dataset (`Gowalla`): HDPView has fewer blocks, leading to noisier results than Privtree for very low-dimensional data. Also, HDPView provides flexible block partitioning. 215
A.5 Number of blocks (log-scale) with various AEs for high-dimensional dataset (i.e., `Adult`) for HDPView and Privtree. HDPView has slightly larger AE blocks, but Privtree has a much more number of blocks i.e., much larger PEs. 216
A.6 HDPView is more effective than Privtree even with controlled number of cuttings on `Numerical-adult` (left) and `Jm1` (log-scale) (right). 216
A.7 Changes in the performance when adding attributes to `Adult` one by one in HDPView, PrivBayes, and HDMM. 218
A.8 The number of blocks generated by HDPView is much lower than that generated by Privtree. 219
A.9 Effects of HDPView's hyperparameter ϵ_r/ϵ_b (Ratio) on `Small-adult` dataset. 221
A.10 γ on `Small-adult`. 221
A.11 β on `Small-adult`. 221

A.12 β on `Numerical-adult` dataset: Optimal β depends on dataset and our default parameter $\beta = 1.2$ is somewhat conservative. . . . 221
A.13 α on `Small-adult`. 221
A.14 λ and δ on various α when $\gamma\epsilon_r = 1.0$. 221
B.1 Trajectory-based PCT overview. 225
B.2 Spatiotemporal Private Set Intersection. 233
B.3 Architecture overview: circled numbers correspond to the steps shown in System Overview. 238
B.4 An overview of running example from raw trajectory to the final risk assessment. 241
B.5 quadkey and QuadKeyEncode. 244
B.6 Mixing TrajectoryHash. 245
B.7 Sequential merge TrajectoryHash. 245
B.8 TrajectoryHash interpretation in the time-series map. 246
B.9 Possible false outcome in terms of the location. 250
B.10 Possible false outcome in terms of time. 250
B.11 *nfp-stPSI* design (2D version for simplicity). 251
B.12 Maximum distance between trajectory point and false positive. . . 251
B.13 Trajectory distributions of New York, Kinki and Tokyo. Note that the latitude and longitude scales are different. 255
B.14 FST can efficiently keep trajectory data encoded by TrajectoryHash for different datasets. 259
B.15 Sizes of 10 chunks for 282 MB server data, and original data size and sum of these chunks. 260
B.16 The execution time varies depending on the number of chunks. . . 260
B.17 The stPSI can achieve high performance on a practical data scale (left). nfp-stPSI causes a larger overhead (right). 261
B.18 Finer granularity of parameters directly degrades the performance. 261
B.19 Results of the stPSI (left) and nfp-stPSI (right) on real data. . . . 261

List of Tables

1.1 Privacy and security properties that can be required by the different example use cases. (✓: Required) 9

1.2 The relationship between FL and two important privacy-preserving technologies, DP and TEE (blue colored), and the proposed frameworks (red colored) in terms of privacy/security properties. (✓: Satisfy, ◗: Partial, ✗: Not) 11

2.1 Comparison with different schemes of DP-FL in terms of trust model and utility. 23

4.1 Datasets and global models in the experiments. 88

4.2 Architectures of the neural networks used in Section 4.4. d is the number of parameters of the global model trained in FL and $|L|$ is the number of labels of inference target. 111

4.3 Architectures of the neural networks used as global models in all FL experiments in Sections 4.4.3 and 4.5.5. Readers can find the details of *ResNet-18* at `https://github.com/weiaicunzai/pytorch-cifar100/blob/master/models/resnet.py`. 112

5.1 MGA can achieve the highest **gains** against all three protocols. $\beta = \frac{M}{N+M}$ and $f_T = \sum_{t \in T} f_t$ in the table. 120

5.2 Overall, output-manipulations are much more vulnerable than input-manipulation. The differences of both manipulations gain are calculated by output-manipulation gain − input-manipulation gain (resp. output-manipulation gain / input-manipulation gain) in Targeted (resp. Untargeted) Attack. 121

A.1 Only the proposed method achieves all requirements in private data exploration for high-dimensional data. Each competitor represents a baseline, data partitioning, workload optimization, and generative model, respectively. 191
A.2 HDPView provides low-error counting queries in average on various workloads and datasets, and high space-efficiency of privacy-preserving materialized view (p-view) when $\epsilon = 1.0$. (N/A is due to HDMM and PrivBayes do not create p-view.) 193
A.3 Datasets. 211
A.4 HDPView's p-view is space efficient (up to $10^{13}\times$). 220
B.1 Comparison with existing approaches. 225
B.2 PSI comparison: cryptography-based(OT, HE) v.s. TEE-based: TEE needs to special hardware, but it has advantages in efficiency and security. 229
B.3 Symbols and parameters. 237
B.4 Approximate scale of the parameter θ_{geo}. 243
B.5 Approximate scale of the parameter θ_{time}. 243
B.6 PSI comparison: cryptography-based vs secure-hardware-based; execution time (balanced) 253
B.7 PSI comparison: cryptography-based vs secure-hardware-based; execution time (unbalanced) 254
B.8 Data set size used in accuracy evaluation. For example, New York's client query data has 100 clients and each of them has 20160 trajectory point records (e.g., every minute for 2 weeks), totally 2016000 records for client query data. 256
B.9 *st-PSI* can achieve high accuracy for all trajectory dataset, but cause false negatives. 257
B.10 *nfp-stPSI* improves accuracy and remove false negatives, but cause false positive. 257

B.11 The method of Reichert et al causes similar errors for sub-sampled NewYork dataset. 258

CHAPTER 1

INTRODUCTION

1.1 Background

In the Big Data era, personal data has been of great value in this decade, leading to its description as *"Personal data is the new oil of the internet and the new currency of the digital world."* [1]. The use of personal data has been very successful from commercial e-commerce services to medical services, and a variety of data analysis tasks have attracted attention, ranging from simple statistical data collection to the recent overwhelming success of machine learning (ML). On the other hand, there have been growing concerns about the privacy of personal data. In particular, the European General Data Protection Regulation (GDPR) [2] has been enforced in the European Union since 2018, and the California Consumer Privacy Act (CCPA) [3] has been enforced in California since 2020. GDPR has already imposed huge fines on some companies (e.g., Amazon [4] and Google [5]) and has had a very large impact on practitioners. Moreover, the damage from data breaches to responsible institutions is increasing year by year [6], which means the cost for institutions to manage personal data continues to increase.

These facts and trends underscore the importance of Federated Learning (FL). FL is an innovative paradigm of privacy-preserving ML that was first introduced by [7]. FL is one of the collaborative machine learning schemes with a single central server, that orchestrates the whole training process, and many clients,

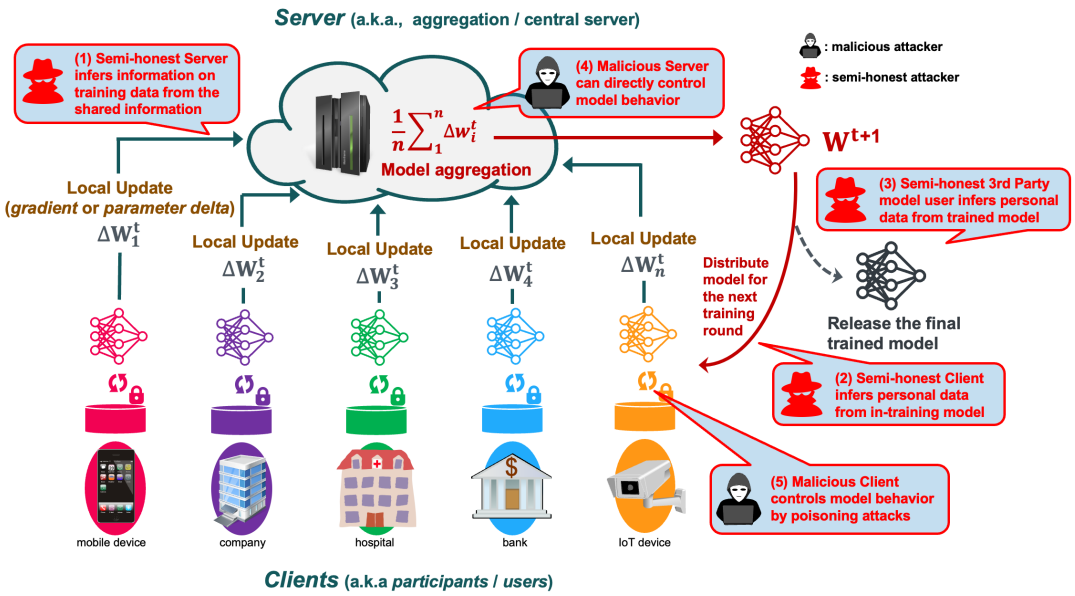


Figure 1.1: Overview of Federated Learning (FL) architecture. A central server (top) that orchestrates the whole training process and many clients (bottom) who have their personal or private data as training data collaboratively train a single global model with local model update for each client. The red balloons indicate the privacy and security risks in FL.

who have their personal or private data as training data, and trains a global model with specific distributed optimization algorithms [8]. Typically, in FL, the server does not need to collect raw data from *clients* (we use *participants* and *users* interchangeably in this thesis)—it only collects *gradients* (or model *parameters delta*) trained on the local data of clients during each round of model training. The server then aggregates the collected gradients from the clients and optimizes the global model with the aggregated updates for each round. Figure 1.1 shows the overview of the typical FL architecture. Thus, FL is expected to enable data analyzers to avoid the expenses and privacy risks of collecting and managing training data containing personal and sensitive information. In addition, by not sharing raw data, it could be possible to analyze data beyond legal and regulatory barriers at the company level [9], hospital level [10], and even at the national level [11, 12]. Therefore, FL has recently attracted a great deal of attention not only from academia but also from industry [13, 14, 15, 16, 17].

Stimulated by the emergence of FL, the term *Federated Analytics* (FA) [18, 19]

has also appeared recently. This refers to a more general and simple data analysis such as collecting telemetry in a federated manner. The term essentially refers to centralized data analysis techniques performed in a distributed setting and without sharing raw data owned by edge clients as well as FL.

Overall, due to its characteristics, FL is one of the promising privacy-enhancing technologies for today's society with high demands for privacy protection.

1.1.1 Privacy and Security problems in FL

FL was originally designed to have higher privacy considerations at the cost of a little degradation of utility due to distributed optimizations. However, many studies have highlighted that the privacy protection of FL is not fully sufficient. On the contrary, being a distributed system, FL has unique security risks compared to traditional centralized ML. Overall, the *privacy risk* is caused by a *semi-honest* attacker (including server, clients, and 3rd party) who tries to obtain extra private information while following the established protocols. The *security risk*, which is described next, is caused by a *malicious* attacker who deviates from the established protocols. In Figure 1.1, the red balloons indicate these privacy and security risks.

Privacy. There are three main privacy risks: privacy can be leaked from (1) gradient information shared to the untrusted server, (2) in-training models shared to the untrusted clients, and (3) trained models released to the untrusted third party.

(1) Gradient information, which is shared from the client to the server for optimization in each training round, is known to be sufficient to leak private information of the client's training data [20, 21, 22, 23, 24]. The gradient information is computed directly from the training data, and it is intuitively understood that it has a strong correlation with the training data. Zhu et al. [20] propose an algorithm called DLG (deep leakage from gradients) to reconstruct the original training data effectively from the shared gradient information, which is the type of attacks called *reconstruction attacks* (a.k.a., *model inversion attacks*). Additionally, many works such as [23, 24] propose inference attacks to reveal sensitive attributes of the target client training data from the gradient information. These results clearly highlight that sharing gradient information performed in FL instead of sharing raw data is in itself far from protecting privacy. This privacy

risk becomes apparent when there are untrusted servers that have access to the raw gradient information.

(2)(3) Regardless of the FL or centralized ML, the trained models memorize training data in their parameters, especially in neural networks [25]. This directly means that a privacy attack is possible if the trained model is accessible by untrusted parties, by *membership inference attacks* [26, 27] and *model inversion attacks* [28, 29]. Membership inference attacks infer samples based on the output of the ML model and attempt to identify whether they are included in the original training data set. For example, by identifying the fact that a clinical record used to train a model is associated with a particular disease, an attacker can infer that the owner of the clinical record has a high probability of suffering from that disease. Model inversion attacks attempt to reconstruct the original training data from the trained model [28]. This privacy risk can also be seen in conventional centralized ML, however, FL is more vulnerable against the risk because many distributed entities can have white-box access to the model in the training phase [27]. Moreover, in a typical application, the trained model will be deployed on the end user's edge device [30, 31], which increases the privacy risk to untrusted 3rd parties. The goal of collaborative learning among multiple organizations can be to share and use the trained model across the organizations [12].

Security. Security risk is not just a violation of the privacy of training data, but rather a subversion of the FL training process that seriously impairs the utility of the trained model or maliciously controls the behavior of the model. The distributed architecture of FL requires multi-party coordination, which is different from the traditional centralized ML scenario, and hence may create FL-specific security risks. The security risks would not be ignored in most cases because untrusted parties can always be present in FL scenarios due to the privacy protection motive. Security risk occurs primarily during model training, by (4) malicious server and (5) malicious clients participating in the training.

(4) In most cases, attacks by the server are limited to the aforementioned privacy risks. This is because the server (here, we do not consider external attacker in outsourced cloud environments and so on) has an incentive to train an accurate and well-behaved model, and will not deviate from the correct protocol. Nevertheless, malicious servers can cause tremendous damage, for example, a malicious server may be able to effortlessly replace a final trained model with another maliciously tuned model. It is not easy for FL participants to always trust the server,

especially when the orchestrator of the FL is another competing company with conflicting interests. Additionally, it has been reported that the server can intentionally control weights and client sampling to achieve more accurate inference attacks against participating clients [32].

(5) Therefore, malicious clients may be more important. They can implant a backdoor into models in FL. The purpose of the backdoor attack is to corrupt the performance of the trained model in certain subtasks (e.g., classifying green cars as frogs). This type of attack is completed through data poisoning [33, 34] or model poisoning [35]. Wang et al. [34] shows poisoning attacks can bypass the defense mechanism such as a truth discovery-based method (e.g., Krum [36]) and effectively succeed in the backdoor attack in FL. Model poisoning has been shown to possibly be more effective than data poisoning [37]. This is also true in FA such as a private frequency estimation task [38, 39, 40]. An attacker can effectively control the output of server-side estimations by bringing some of the clients under his control. This is also known as Byzantine Attacks where an attacker injects some fake clients among the participants and also executable in FL [41].

1.2 Differential Privacy

Differential Privacy (DP) [42] is a rigorous mathematical privacy definition that quantitatively evaluates the degree of privacy protection when publishing statistical outputs such as ML models. The basic principle is to provide indistinguishability to the input database by appropriately randomizing the output (e.g., adding Laplace noise to the histogram, etc.). DP has become the de facto standard for statistical privacy protection methods because it provides worst-case privacy guarantees against arbitrary attacks, which differs from classical anonymization-based methods, including k-anonymity. Over the past few years, DP has also attracted industry attention [43, 44, 45, 46] and has been used at the US Census [47].

An important recent application of DP is ML. DP-SGD, proposed by Abadi et al. [48], allows general trained models to be differentially private by interspersing gradient randomization into the optimization step. This results in ML models with interpretable and quantifiable privacy guarantees. The differentially private ML technique is important in scenarios where the trained models, including only

access to the model inference, are released to untrusted parties different from the owner of the training data. The technique has been progressively successful with a tight analysis of privacy loss bounds [49, 50, 51] and has also been tested empirically against practical privacy attacks [52, 53].

DP is often combined with FL, which is called differentially private FL (DP-FL) [54, 55, 56, 57]. Researchers have explored several DP-FL variants, such as using central DP (CDP-FL) [58], local DP (LDP-FL) [56], and Shuffle DP (Shuffle DP-FL) [57], to strike a good balance between privacy and utility. Note that they have different trust models. In CDP, which is the original DP definition, data owners trust the server, and the server acts as a data curator to collect the raw data. On the other hand, in LDP, data owners do not need to trust the server since the data must be randomized on the client side before being collected. This is more advantageous than CDP in terms of the trust model; however, it requires even stronger perturbation and sacrifices utility significantly. To overcome the weakness of the utility of LDP by privacy amplification, a method using the shuffler model has been proposed recently [59, 60, 61], which is called Shuffle DP. And it has been applied to FL [57].

1.3 Trusted Execution Environment

Trusted Execution Environment (TEE), as defined formally in [62][a], creates an isolated execution environment within untrusted computers (e.g., cloud VMs), which enables securing computation that can ensure data confidentiality and/or integrity throughout the whole computation against potential attackers. The general TEE security model provides security against privileged software, including operating systems and hypervisors, by using special hardware as a root-of-trust. In addition to guarantees of confidentiality and integrity, the important features of TEE are the ability to perform arbitrary computation and native-like performance. These are because TEE decrypts encrypted data and executes CPU instructions on plain text within the trusted CPU packages. Because of these appealing characteristics, TEE has attracted attention as a next-generation security technology.

TEE can be seen as an alternative to other software-based Multi Party Compu-

[a]There can be a similar but another popular definition, e.g., by the Global Platform [63], which focuses more on the system aspects.

tation (MPC) techniques. For example, cryptographic techniques such as Homomorphic Encryption (HE) [64], Oblivious Transfer (OT) [65], or Secure Aggregation (SA) [66] may achieve equivalent functionality without the special hardware assumption of TEE. For example, the recent paper [67] categorized TEE as one of the methods of SA, and [68] consider server-side verifiability with binding commitments. However, they are vastly inferior in terms of efficiency due to the heavy computation costs. In other words, it can be interpreted that there is a trade-off between hardware requirements and efficiency to achieve security. The general acceptance of having special hardware is a subtle issue. Currently, one answer could be the use of TEE on the server side is acceptable [69], but on the client side (e.g., mobile devices) would be a somewhat stronger assumption, especially for powerful TEE such as Intel SGX [70]. Therefore, it may be necessary to consider the implementation of conventional MPC techniques instead of simply using TEE, particularly on the client side.

Enhancing FL using TEE is a promising approach to achieve private and secure FL, which has garnered significant academic attention in recent years [71, 72, 73, 74, 75]. TEE provides confidentiality, integrity, and functionalities such as Remote Attestation for verifiability, fully justifying its use on the untrusted server side in FL [72, 76, 74]. After the first verification of the remote TEE, the shared gradients can be transmitted to the TEE via a secure channel and computed securely in a confidential, verifiable, and arbitrary manner, thereby eliminating some of the aforementioned attack surfaces on the server side.

1.4 Integration into FL

DP and TEE are very important techniques in the next-generation privacy-preserving data analysis from the statistical and cryptographic approaches, respectively. The belief is that these technologies can be the key to solving the privacy and security problems in FL as described in Section 1.1.1. Therefore, the goal of this thesis is to answer the following research question: *To enhance privacy and security in FL, how can the advanced privacy-preservation techniques such as DP and TEE be effectively integrated into FL?* To this end, a set of appropriately abstracted key privacy/security properties covering all the risks mentioned earlier, i.e., (1)-(5) in Figure 1.1, is established.

The following five important privacy/security properties **A**-**E** in FL correspond

to the above-mentioned risks (1)-(5), respectively:

A. **Privacy against Central Server**: the privacy risk for semi-honest central server who can access the intermediate information.

B. **Privacy against Clients**: the privacy risk for semi-honest clients who have access to the in-training model in training rounds.

C. **Privacy against 3rd Party**: the privacy risk for semi-honest external users who have access to the final trained model.

D. **Security against Central Server**: the security risk for malicious server who can control the training process and manipulate the intermediate model. (Note that, in many cases, the server is the same as the FL administrator and therefore has no motivation to deviate from the established FL protocol, and this may not be as serious in practice.)

E. **Security against Client**: the security risk for malicious client who can access and control the in-training model locally in the training rounds.

The need for each of these privacy/security properties depends entirely on the trust model assumed by the target use cases. While **B** is specific to FL, the remaining properties are found in existing common data analysis among untrusted parties. To facilitate understanding, several real-world scenarios are illustrated with examples as shown in Table 1.1.

(a) The first scenario is census, where governments must meet both the obligation to release statistical information to the public and the obligation to protect the privacy of people providing the personal data. While the statistical information will be publicly available and thus needs to be properly privatized (i.e., **C**), a large number of malicious clients who provide faked information may intentionally manipulate the estimated statistics (i.e., **E**) because, for example, government budget allocations or other critical decisions are made based on the census results.

(b) The second scenario is internal data analytics of an E-commerce site, where the EC site operator wants to analyze the individual history data to design the marketing strategy or improve the service. Analytical results will not be released to the public but could be seen by the service provider. Therefore, conservative clients may not allow their individual-level behavior, such as purchase history, to be known to the service provider (i.e., **A**). (c) The item recommendation results often displayed on EC sites are public information. If the predictive model is

Table 1.1: Privacy and security properties that can be required by the different example use cases. (✓: Required)

	A. Privacy against Central Server	*B. Privacy against Clients*	*C. Privacy against 3rd Party*	*D. Security against Central Server*	*E. Security against Client*
(a) Census	-	-	✓	-	✓
(b) EC site (Internal data analytics)	✓	-	-	-	-
(c) EC site (Recommendation page)	✓	-	✓	✓	✓
(d) MLaaS (Inference API)	✓	-	-	✓	-
(e) Sticker recommendation with FL	✓	✓	✓	✓	✓

trained from the user's personal data, in the worst case, the information based on the inference results may lead to a violation of training data privacy (i.e., **C**). Also, users may not want the server to know their personal data used for training the predictor (i.e., **A**). Servers may intentionally post false ranking results while claiming fairness and to be based on correct collected data (i.e., **D**). For some interests, external attackers may intentionally generate false behavioral history data to control the results of the result (i.e., **E**), for example, a company wants to make sure that many of its items are recommended.

(d) Considering ML as a Service (MLaaS), which hosts trained models and opens the inference interface, a user with private input data would send the input to the server and get only inference results. (Also, there could be fine-tuning.) At this point, the user may want to hide the sensitive input data from the untrusted service provider (i.e., **A**), and the service provider may deliberately not compute the inference correctly (i.e., **D**), e.g., for saving resources.

(e) Based on the above, lastly, let's look at some examples of FL already in

industrial operation. There is already an example of FL in actual use for stamp recommendation in a messaging application [15]. This is a typical example of a scenario in which end-users of the application participate in FL as clients, and FL trains a global model that recommends stamps to be used in messages based on the context of the message. The more a stamp is used, the more the creator receives an incentive. After the model training, the global model is distributed to each end user's device and used on the edge devices. Here, the users want to hide their data from the server since their private message information is used in the model (i.e., **A**). The model will be accessible to the end user during and after training, creating a privacy risk against other participants (i.e., **B, C**). Since the model is trained on the end-user's device using the end-user's local data, end-users who want to increase their stamp sales have an incentive to intentionally tamper with the process to get their stamps recommended (i.e., **E**). Similarly, the server may have an incentive to intentionally tamper with the model training process so that a particular stamp is highly recommended (i.e., **D**).

Relationships with FL. Table 1.2 shows the relationships between FL and the two important privacy-preserving technologies, DP and TEE, and our proposed three federated frameworks in terms of the privacy/security properties. All of the privacy/security properties are missing from plain FL (top of the Table). The following descriptions focus primarily on how each property lacked in FL is complemented by DP, TEE, and the alternative methods.

A. **Privacy against Central Server**: Server-side TEE or MPC and LDP can be used to hide the intermediate information of FL (e.g., gradient, model parameters delta) from the semi-honest central server to prevent privacy leaks.

B. **Privacy against Clients**: DP can be used to privatize the in-training model to prevent privacy leaks.

C. **Privacy against 3rd Party**: DP can complement FL to privatize the final output of the whole data analysis process (estimated statistics, trained model) with formal privacy guarantees. It enables the release of the output to third parties.

D. **Security against Central Server**: Server-side TEE can prevent the malicious central server from manipulating the training process and model behavior directly.

Table 1.2: The relationship between FL and two important privacy-preserving technologies, DP and TEE (blue colored), and the proposed frameworks (red colored) in terms of privacy/security properties. (✓: Satisfy, ◗: Partial, ✗: Not)

	A. Privacy against Central Server	*B. Privacy against Clients*	*C. Privacy against 3rd Party*	*D. Security against Central Server*	*E. Security against Client*
FL	✗	✗	✗	✗	✗
DP (e.g., [77])	CDP ✗ LDP ✓	✓	✓	✗	✗
TEE (or MPC) (e.g., [78])	✓	✗	✗	✓	✓
ULDP-FL [79] (FL+DP(+MPC))	✗ (+MPC ✓)	✓	✓	✗	✗
OLIVE [24] (FL+TEE(+DP))	✓	✗ (+DP ✓)	✗ (+DP ✓)	✓	✗
Verifiable LDP [40] (FA+LDP+MPC)	✓	✓	✓	✗	◗

E. **Security against Client**: Client-side TEE or MPC can protect FL training processes from poisoning attacks by malicious clients to control and/or downgrade model behavior.

Thus, it can be seen that DP and TEE or alternative MPC can partially complement the properties that are missing in the plain FL. The need for each of these five properties depends fully on the actual FL application scenario. The next section outlines our three proposed privacy-security enhanced FL frameworks that work for each specific scenario.

1.5 Contributions

Therefore, FL applications have different requirements depending on the scenario. In this thesis, we studied the following three proposed frameworks that can integrate DP and/or TEE or alternative MPC into FL to satisfy different requirements for different purposes, as shown in Table 1.2. We solve cutting-edge challenges specific to each scenario in each of the following works, as outlined in Sections 1.5.1, 1.5.2 and 1.5.3. Finally, we discuss the social impact of the proposed private and secure FL frameworks by showing the real-world FL application scenarios.

1.5.1 Across-silo User-level DP in cross-silo FL

In Chapter 3, we focus on practical DP-FL which has garnered attention [54] to train a theoretically private model in FL. The model with guaranteed DP achieves **B** and **C**. The original definition of DP assumes a single record as the unit of privacy, which does not give meaningful privacy in a general setting with multiple records that belong to a single user. It motivates the definition of user-level DP [80, 81, 82, 83], which guarantees indistinguishability for all records held by a user instead of a single record. However, it is still unclear how user-level DP can be guaranteed in a general cross-silo FL setting where a single user's data exists across silos.

We present ULDP-FL, a novel FL framework designed to guarantee user-level DP in cross-silo FL where a single user's data may belong to multiple silos. Our proposed algorithm directly ensures user-level DP through per-user weighted clipping, departing from group-privacy approaches. We provide a theoretical analysis of the algorithm's privacy and utility. Additionally, we enhance the algorithm's utility by optimized weighting method and showcase its private implementation using cryptographic building blocks. Empirical experiments on real-world datasets show substantial improvements in our methods in privacy-utility trade-offs under user-level DP compared to group-privacy-based baseline methods. To the best of our knowledge, our work is the first FL framework that effectively provides user-level DP in the general cross-silo FL setting.

1.5.2 FL on TEE against the risk of sparsification

In Chapter 4, we consider combining FL with a server-side TEE, which is a promising approach for realizing privacy-security enhanced FL complementing **A** and **D**. This combination has garnered significant academic attention in recent years [71, 72, 73, 74, 75]. Implementing the TEE on the server side enables each round of FL to proceed without exposing the client's gradient information to untrusted servers. This addresses usability gaps in existing secure aggregation schemes [66] as well as utility gaps in DP-FL [57]. However, to address the issue using a TEE, the vulnerabilities of server-side TEEs that reveal memory access patterns [84, 85, 86] need to be considered — this has not been sufficiently investigated in the context of FL.

The main technical contribution of this study is the analysis of the vulnerabilities of server-side TEE in FL and its defense. First, we theoretically analyze the leakage of memory access patterns, revealing the risk of sparsified gradients, which are commonly used in FL to enhance communication efficiency and model accuracy. Second, we devise an inference attack to link memory access patterns to sensitive information in the training dataset. Finally, we propose an efficient yet oblivious ([87]) aggregation algorithm to prevent memory access pattern leakage. The experiments on real-world data demonstrate that the proposed method functions efficiently in practical scales.

1.5.3 Verifiable LDP protocol against Untrusted Client

In Chapter 5, as a first step in defending against malicious clients in FL, i.e., **E**, we begin the initial research with one of the simplest federated analytics settings, frequency estimation under LDP. Basic LDP algorithms such as Randomized Response have been well studied with a focus on improving the utility. However, recent studies show that LDP is generally vulnerable to malicious data providers in nature [39, 38] as well as FL. Because a data collector has to estimate background data distribution only from already randomized data, malicious data providers can manipulate their output before sending, i.e., randomization would provide them plausible deniability. Attackers can skew the estimations effectively since the calculation assumes that there is a randomization for LDP which can be used as leverage for attacks, and can even control the estimations.

We show how to prevent malicious attackers from compromising LDP protocol.

Our approach is to utilize a verifiable randomization mechanism. The data collector can verify the completeness of executing an agreed randomization mechanism for every data provider. The proposed method completely protects the LDP protocol from output-manipulations, and significantly mitigates the expected damage from attacks. We describe the secure version of three state-of-the-art LDP protocols and empirically show they cause acceptable overheads according to several parameters. We believe this type of client-side verifiability that we propose in this method can be extended to prevent attacks in more complicated federated tasks including FL in the future.

CHAPTER 2

Literature Review

2.1 Federated Learning

Federated learning (FL) [7, 88] is a recent ML scheme based on distributed optimization with a single central aggregation server and many participants who have their own private training data. The basic FL algorithm, called FedAVG [7, 89], trains models by repeating model optimization steps in the local environment of the participants and updating the global model by aggregating the parameters of the locally trained models. FedSGD [7] exchanges locally updated gradients based on distributed stochastic gradient descent. Overall, users are not required to share their training data with the server, which represents a major advantage over traditional centralized ML.

In each round, the server aggregates models from all participants and then redistributes the aggregated models. Each participant $u \in U$ optimizes a local model f_u, which is the expectation of a loss function $F(x;\xi)$ that may be non-convex, where $x \in \mathbb{R}^d$ denotes the model parameters with d model dimension and ξ denotes the data sample, and the expectation is taken over local data distribution $\mathcal{D}_u$. We optimize this global model parameter cooperatively across all participants. Formally, the overarching goal in FL can be formulated as follows:

$$\min_x \left\{ f(x) := \frac{1}{|U|} \sum_{u \in U} f_u(x) \right\}, \; f_u(x) := \mathbb{E}_{\xi \sim \mathcal{D}_u} F(x;\xi). \tag{2.1}$$

Algorithm 1 FedAVG [7, 89]

Input: U: participants, q: sampling rate, η_l, η_g: learning rate for local and global, E: #local epochs

1: **procedure** TRAIN(U, q, η_l, η_g)
2: Initialize model x_0
3: **for** each round $t = 0, 1, 2, \ldots$ **do**
4: $\mathcal{Q}^t \leftarrow$ poisson sub-sampling users from U for round t with sampling rate q ▷ Participant Selection: select participants for round t
5: **for** each user $u \in \mathcal{Q}^t$ **in parallel do**
6: $\Delta_u^t \leftarrow \text{CLIENT}(x_t, \eta_l, E)$ ▷ Broad Cast and Local Training: Participant u downloads the global model and computes parameter delta
7: **end for**
8: $\tilde{\Delta}^t = \frac{1}{q|U|} \sum_{u \in \mathcal{Q}^t} \Delta_u^t$ ▷ Aggregation: average model parameter delta
9: $x_{t+1} \leftarrow x_t + \eta_g \bar{\Delta}^t$ ▷ Model Update: update global model
10: **end for**
11: **end procedure**
12: **procedure** CLIENT(x_t, η, E)
13: $x \leftarrow x_t$
14: **for** each epoch $e = 0, 1, 2, \ldots, E$ **do**
15: $\mathcal{G} \leftarrow$ user i's local data split into batches
16: **for** batch $g \in \mathcal{G}$ **do**
17: $x \leftarrow x - \eta \nabla F(x; g)$ ▷ Stochastic gradient descent (SGD)
18: **end for**
19: **end for**
20: $\Delta \leftarrow x - x_t$ ▷ Compute model parameter delta
21: **return** Δ
22: **end procedure**

In round $t \in [T]$ in FL, the global model parameter is denoted as x_t. The algorithm description of FedAVG is shown in Algorithm 1, which includes mainly repeated five steps: participant selection, broadcast, local training, aggregation, and model update. Compared to conventional simple distributed SGD such as FedSGD, which exchanges a one-time stochastic gradient, FedAVG exchanges differences in the parameters of models that are trained by some local epoch (Lines 14 - 20), resulting in faster convergence [89] and reducing communication costs, which is likely to be a practical bottleneck in FL [90, 91, 92].

In FL, model training is only one part of the whole, with many steps of the ML pipeline including model design, hyperparameter tuning, building the training infrastructure, evaluating the model, and deploying the model [93]. These often include FL-specific challenges [17]. For example, the cost of tuning for model architectures and hyperparameters is often prohibitively high [94], and the lack of access to raw data during model evaluation makes it difficult to detect bias in the training data set, such as outliers [95].

General Research Points

The main research points of FL are as follows:

- **Privacy and Security**: As described in Section 1.1.1, FL lacks in privacy against semi-honest attacker and in security against malicious attacker, which causes privacy risks even without sharing raw data, which is crucial given the main goal of FL and we mainly focus on in this thesis. In particular, Secure Aggregation (SA) [67] is a popular parameter aggregation method in private FL for concealing individual parameters. It is based on the lightweight pairwise-masking method [96, 66, 97], homomorphic encryption [98, 99] or TEE [74, 71].
- **Utility**: There is a significant degradation of model performance and convergence speed [89, 100], which can be studied mainly in the context of distributed optimization for heterogeneous data distribution (non-iid) [101, 102].
- **Communication Efficiency**: Communication costs tend to be a bottleneck in FL systems because model parameters, which may be deep models, need to be exchanged many times [90, 91, 92]. Hence, parameter sparsification is important in a practical scenario [97, 103].
- **Developability**: Compared to centralized ML, FL requires a considerably more complicated infrastructure due to its decentralized setup, especially in security, availability, and device heterogeneity aspects, which can be an obstacle for practitioners. Several popular implementations have been developed, such as Google TensorFlow Federated (TFF), FedML [104], and PySyft [105], etc.

Cross-device and Cross-silo FL

There are two variants of FL, called *cross-device FL* and *cross-silo FL*, depending on whether the participants are individual edge devices (such as smartphones) or organizations (such as hospitals), respectively [93]. Typically, in cross-device FL [7, 54, 96], the number of participants is large (e.g., up to 10^{10}), and the participants are probabilistically selected to participate in each round of training. On the other hand, in cross-silo FL [106, 102, 107, 108], the number of participants is small (e.g., 2-100), and all participants participate in all rounds of training. Also, they differ in various aspects such as accessible computing resources, communication infrastructure, data distribution and partitioning (*horizontal* or *vertical*), trust models, and incentive designs.

Real-world deployment

FL has recently attracted a great deal of attention not only from academia but also from industry. For example, it has been continuously tested in Google's productions [31, 30, 109, 13]. Recently, platformers such as Meta [14], Apple [17], and LinkedIn [16] have followed Google's early research and focused their efforts. Additionally, in Japan, LINE has deployed a model trained using FL for sticker recommendation in its app [15]. However, even formal legal rules for privacy have not yet been fully developed, and it is likely that these are only a trial introduction for the purpose of testing technical perspectives.

2.2 Differential Privacy

Differential Privacy (DP) [42] is a rigorous mathematical privacy definition that quantitatively evaluates the degree of privacy protection when outputs are published. The importance of DP is underscored by the fact that the US census announced '2020 Census results will be protected using "differential privacy", the new gold standard in data privacy protection' [47]. As shown in the following definition, in DP, the degree of privacy is parameterized by ϵ. This is also referred to as the *privacy budget*: the larger the ϵ, the looser the privacy bounds allowed, and the smaller the ϵ, the stronger the privacy protection. δ represents the failure probability of the privacy guarantee, which usually needs to be small enough to be proportional to the database size.

Definition 1 ((ϵ, δ)-DP)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (ϵ, δ)-DP if, for any two input databases $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most one record and any subset of outputs $Z \subseteq \mathcal{Z}$, it holds that*

$$\Pr[\mathcal{M}(D) \in Z] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in Z] + \delta. \tag{2.2}$$

In the special case where $\delta = 0$, it is simply called ϵ-DP and in cases where $\delta > 0$, it is sometimes referred to as *approximate DP*. Databases D and D' are called *neighboring* databases. The maximum difference in the output for any neighboring database is referred to as *sensitivity*, as defined in Definition 2. Practically, we employ a randomized mechanism $\mathcal{M}$ that ensures DP for a function f. The mechanism $\mathcal{M}$ perturbs the output of f to cover f's sensitivity.

Definition 2 (Sensitivity)**.** *The sensitivity of a function f for any two neighboring inputs $D, D' \in \mathcal{D}$ is:*

$$\Delta_f = \sup_{D, D' \in \mathcal{D}} \|f(D) - f(D')\|.$$

where $\|\cdot\|$ is a norm function defined in f's output domain.

Given a statistical query f, we can create a DP mechanism $\mathcal{M}$ by analyzing the sensitivity Δ_f of the query f and designing the randomization accordingly. For example, the *Laplace mechanism* [42] is well-known as a standard approach for randomizing numerical outputs such as counting queries, which satisfies ϵ-DP.

Definition 3 (Laplace Mechanism)**.** *For function $f : \mathcal{D} \to \mathbb{R}^n$, the Laplace mechanism adds noise to $f(D)$ as:*

$$f(D) + Lap(\Delta_f/\epsilon)^n. \tag{2.3}$$

where $Lap(\lambda)^n$ denotes a vector of n independent samples from a Laplace distribution $Lap(\lambda)$ with mean 0 and scale λ.

Quantifying the privacy of differentially private mechanisms over multiple releases is essential for releasing multiple outputs. This is called *composition* for privacy consumption and is one of the most important properties of DP. Sequential composition and parallel composition are standard privacy accounting methods for multiple releases.

Theorem 1 (Sequential Composition [42]). *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be mechanisms satisfying ϵ_1-, $\ldots, \epsilon_k$-DP. Then, a mechanism sequentially applying $\mathcal{M}_1, \ldots, \mathcal{M}_k$ satisfies $(\sum_{i\in[k]} \epsilon_i)$-DP.*

Theorem 2 (Parallel Composition [110]). *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be mechanisms satisfying ϵ_1-, $\ldots, \epsilon_k$-DP. Then, a mechanism applying $\mathcal{M}_1, \ldots, \mathcal{M}_k$ to disjoint databases $D_1, \ldots, D_k$ in parallel satisfies $(\max_{i\in[k]} \epsilon_i)$-DP.*

However, recent works have developed many tighter composition theorems on approximate DP by using basic probability theory techniques to bound the sum of probability distributions of the outputs of the DP mechanisms, such as *Advanced Composition* [111] and *R'enyi DP (RDP)* [50, 49], etc.

RDP is a variant of approximate DP based on R'enyi divergence. It is preferred over other composition theorems because it is easy to use for the *Gaussian mechanism* [50] and has a tighter bound than the standard composition theorems. The following lemmas give the bounds of the RDP for a typical mechanism and are used to further convert it to an original DP bound, which we use in this thesis. In particular, as described in Lemma 4, applying random sub-sampling to the input data before applying the supposed randomized mechanism can greatly improve the privacy bound. This is called *privacy amplification* by sub-sampling.

Definition 4 ((α, ρ)-RDP [50]). *Given a real number $\alpha \in (1, \infty)$ and privacy parameter $\rho \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies (α, ρ)-RDP if for any two neighboring datasets $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most one record, we have that $D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) \leq \rho$ where $D_\alpha(\mathcal{M}(D)||\mathcal{M}(D'))$ is the Rényi divergence between $\mathcal{M}(D)$ and $\mathcal{M}(D')$ and is given by*

$$D_\alpha(\mathcal{M}(D)||\mathcal{M}(D')) := \frac{1}{\alpha - 1} \log \mathbb{E}\left[\left(\frac{\mathcal{M}(D)}{\mathcal{M}(D')}\right)^\alpha\right] \leq \rho,$$

where the expectation is taken over the output of $\mathcal{M}(D)$.

Lemma 1 (RDP composition [50]). *If $\mathcal{M}_1$ satisfies (α, ρ_1)-RDP and $\mathcal{M}_2$ satisfies (α, ρ_2), then their composition $\mathcal{M}_1 \circ \mathcal{M}_2$ satisfies $(\alpha, \rho_1 + \rho_2)$-RDP.*

Lemma 2 (RDP to DP conversion [112]). *If $\mathcal{M}$ satisfies (α, ρ)-RDP, then it also satisfies (ρ', δ)-DP for any $0 < \delta < 1$ such that*

$$\rho' = \rho + \log \frac{\alpha - 1}{\alpha} - \frac{\log \delta + \log \alpha}{\alpha - 1}.$$

Lemma 3 (RDP Gaussian mechanism [50]). *If $f : D \to \mathbb{R}^d$ has $\ell 2$-sensitivity Δ_f, then the Gaussian mechanism $G_f(\cdot) := f(\cdot) + \mathcal{N}(0, I\sigma^2\Delta_f^2)$ is $(\alpha, \alpha/2\sigma^2)$-RDP for any $\alpha > 1$.*

Lemma 4 (RDP for sub-sampled Gaussian mechanism [49]). *Let $\alpha \in \mathbb{N}$ with $\alpha \geq 2$ and $0 < q < 1$ be a sub-sampling ratio of sub-sampling operation $Samp_q$. Let $G'_f(\cdot) := G_f \circ Samp_q(\cdot)$ be a sub-sampled Gaussian mechanism. Then, G'_f is $(\alpha, \rho'(\alpha, \sigma))$-RDP where*

$$\rho'(\alpha, \sigma) \leq \frac{1}{\alpha - 1} \log\left(1 + 2q^2 \binom{\alpha}{2} \min\left\{2(e^{1/\sigma^2} - 1), e^{1/\sigma^2}\right\} + \sum_{j=3}^{\alpha} 2q^j \binom{\alpha}{j} e^{j(j-1)/2\sigma^2}\right).$$

In general, we can compute tighter numerical bounds for these bounds, in addition to the closed-form upper bounds described above [49, 51].

Fundamental researches. Thanks to these attractive properties, DP has been applied to a wide range of private data analysis tasks, from releasing range counting queries [113, 114, 77] to machine learning [48, 37].

Private counting queries, such as observing the histogram of the private data, is a fundamental and important task in private data analysis. In [77], we study how to construct an explorable privacy-preserving materialized view under DP, which can release unlimited range counting queries. No existing state-of-the-art methods simultaneously satisfy the following essential properties in data exploration: workload independence, analytical reliability (i.e., providing error bound for each search query), applicability to high-dimensional data, and space efficiency. To solve the above issues, we propose HDPView, which creates a differentially private materialized view by well-designed recursive bisected partitioning on an original data cube, i.e., count tensor. The proposed method searches for block partitioning to minimize the error for the counting query, in addition to randomizing the convergence, by choosing the effective cutting points in a differentially private way, resulting in a less noisy and compact view. Details of this work are presented in the Appendix A.

Differentially private ML is a promising approach to protect privacy in ML. The most significant breakthrough is DP-SGD (shown in Algorithm 2) and its privacy accounting *moments accountant* [48]. This method guarantees DP for the trained model by adding Gaussian noise on the stochastic gradient in the

Algorithm 2 DP-SGD [48]

Input: C: clipping bound, σ: noise multiplier, q: sampling rate, N: #data, η: learning rate

1: **procedure** TRAIN(C, q, η)
2: Initialize model x_0
3: **for** each iteration $t = 0, 1, 2, \ldots$ **do**
4: Take a random sample (data) $\mathcal{L}^t$ with sampling rate q
5: **for** each sample $i \in \mathcal{L}^t$ **do**
6: Compute stochastic gradients $g_{t,i}$ ▷ $\mathbb{E}[g_{t,i}] = \nabla F(x_t, i)$
7: $\tilde{g_{t,i}} \leftarrow g_{t,i} \cdot \min\left(1, \frac{C}{\|g_{t,i}\|_2}\right)$ ▷ clipping with C
8: **end for**
9: $\tilde{g}_t = \frac{1}{qN}\left(\sum_{i \in \mathcal{L}^t} g_{t,i} + \mathcal{N}(0, \mathbf{I}\sigma^2 C^2)\right)$ ▷ Noise addition
10: $x_{t+1} \leftarrow x_t - \eta \tilde{g}_t$ ▷ Gradient Descent
11: **end for**
12: **Output** final x_T ▷ T may be given or determined by privacy budget consumption or model convgergence
13: **end procedure**

optimization step (Line 9). The important point is that each stochastic gradient is computed for each sample and clipped by given *clipping bound* to bound the sensitivity of the gradient (Line 7). In addition, sub-sampling of the data (Line 5) leads to a tighter privacy bound by *privacy amplification by sub-sampling* [49]. The combination of DP and ML has been extensively studied in terms of privacy attacks on trained models [37, 26] and in terms of differentially private generative models that synthesize private data [115, 116, 117].

2.2.1 Differentially Private Federated Learning

Differentially private FL (DP-FL) [58, 54] has garnered significant attention due to its capacity to alleviate privacy concerns by ensuring DP. Researchers have explored various DP-FL techniques to strike a good balance between trust models and utility, as shown in Table 2.1.

In central DP FL (CDP-FL) [58, 54, 118, 55], a trusted server collects raw participants' data and takes responsibility for privatizing the global model. Typically, cross-device FL often considers client-level DP [54] rather than the original record-level DP because each client (device) can have multiple records. (Client-

Table 2.1: Comparison with different schemes of DP-FL in terms of trust model and utility.

	Trust model	Utility
CDP-FL [58, 54, 118, 55]	Trusted server	Good
LDP-FL [56, 119, 120, 121]	Untrusted server	Limited
Shuffle DP-FL [122, 123, 57]	Untrusted server + Shuffler	$\leq$ CDP-FL
CDP-FL with TEE [24]	Untrusted Server with TEE	= CDP-FL

level) CDP-FL guarantees that it is probabilistically indistinguishable whether a client is participating in the training or not. It is defined as follows:

Definition 5 ((client-level) (ϵ, δ)-DP [54])**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (ϵ, δ)-DP if, for any two neighboring datasets $D, D' \in \mathcal{D}$ such that D' differs from D in at most one client's record set and any subset of outputs $Z \subseteq \mathcal{Z}$, it holds that*

$$\Pr[\mathcal{M}(D) \in Z] \leq \exp(\epsilon) \Pr[\mathcal{M}(D') \in Z] + \delta.$$

Where $\mathcal{Z}$ corresponds to the final trained model and $\mathcal{M}(D)$ corresponds to the training algorithm with perturbation (e.g., DP-SGD) that uses input client D's training data to learn. In general, CDP-FL provides a good trade-off between privacy and utility (e.g., model accuracy) of differentially private models even at practical model scales [54, 118]. ULDP-FL [79] has studied the methodology for deploying this client-level CDP guarantee in a general cross-silo FL setting where a single user's multiple records can exist across silos, as introduced in Chapter 3. However, CDP-FL requires the server to access raw gradients, which leads to major privacy concerns on the server as the original data can be reconstructed even from the raw gradients [20, 21].

In LDP-FL [56, 119, 120, 121], the clients perturb the gradients before sharing with an *untrusted* server, guaranteeing formal privacy against both malicious third parties and the untrusted server. LDP-FL does not require a trustful server unlike CDP-FL. However, LDP-FL suffers from poor privacy-utility trade-offs, especially when the number of users is not sufficient (i.e., the signal is drowned in noise) or the number of model parameters is large (i.e., more noise is needed for achieving the same level of DP). Unfortunately, it is limited to models with an extremely small number of parameters or companies with a huge user base (e.g., 10 million).

To overcome the weakness of the utility of LDP by privacy amplification, a method using the *shuffler model* [124, 59], has been proposed [122], i.e., Shuffle DP-FL. This method introduces a trusted shuffler instead of trusting the server and achieves some level of utility. However, it cannot outperform CDP in utility because we can simulate the shuffling mechanism on a trusted server. The privacy amplification of the shuffler also has weaknesses, such as the need for a large number of participants and small parameter size due to the underlying LDP limitation. This is clearly highlighted in Table 12 of [57] [a]. Hence, there is still a utility gap between CDP-FL and the state-of-the-art Shuffle DP-FL.

To fill this gap, our proposed Olive [24] employs TEE to ensure secure model aggregation on an untrusted server so that only differentially private models are observable by the untrusted server or any third parties. The details of Olive are presented in Chapter 4. As shown in Table 2.1, the utility of Olive is exactly the same as the conventional CDP-FL as the computation inside TEE can be implemented for arbitrary algorithms. Note that there are differences from the pairwise-masking secure aggregation, which has limitations on the DP mechanism. For example, it requires discretizing the parameters and noises and adding noises in a distributed manner [96, 125].

2.2.2 Differentially Private Federated Analytics

In addition to FL, Federated Analytics (FA) [19] basically use LDP, allowing federated and private analysis of relatively simple statistics [126, 127]. Although FA with DP is a promising approach to privacy-preserving data analysis [19], it is still in its infancy and has not been fully investigated in terms of security. For example, they are not intended for malicious clients. Recent works [39, 38] have suggested that the LDP protocol used in the FA can increase gains of the malicious client-side attackers, and defensive measures against this need to be considered. To fill this defensive gap, we propose a verifiable randomization mechanism to prevent the manipulation of LDP protocols by malicious clients [40] as introduced in Chapter 5.

[a] We can reproduce a similar result with our code `https://github.com/FumiyukiKato/FL-TEE/blob/master/src/eval-ldp-sgd.py`.

2.3 Trusted Execution Environment

Trusted Execution Environment (TEE) uses special hardware features to create a secure execution environment in an untrusted remote machine, guaranteeing *confidentiality*, *integrity*, and *verifiability* [62]. Simply put, TEEs enable secure computation externally while avoiding computationally expensive encryption methods such as Homomorphic Encryption (HE) [64], Oblivious Transfer (OT) [65], and others. Several implementations of TEE have been announced by industry (e.g., Intel SGX [70], Arm TrustZone [128], and AMD SEV [129]) and academia (e.g., Sanctum [130], KeyStone [131]). They have different CPU modes and instruction sets.

We focus on a well-known TEE implementation—Intel SGX [70], which is currently the most prominent and commercially available choice of TEE. In particular, SGX supports production-ready *Remote Attestation* service [132]. SGX is an extended instruction set for Intel x86 processors, enabling the creation of an isolated memory region called an *enclave*. The enclave resides in an encrypted and protected memory region called an *EPC*. The data and programs in the EPC are transparently encrypted outside the CPU package by the Memory Encryption Engine, enabling performance comparable to native performance. SGX assumes the CPU package to be the trust boundary—everything beyond it is considered untrusted—and prohibits access to the enclave by any untrusted software, including the OS/hypervisor. Note that for design reasons, the user-available size of the EPC is limited to approximately 96 MB for most current machines. When memory is allocated beyond this limit, SGX with Linux provides a special paging mechanism. This incurs significant overhead for encryption and integrity checks, resulting in poor performance [78, 133, 134].

Attestation. SGX supports Remote Attestation (RA), which can verify the correct initial state and genuineness of an enclave. This prevents malicious attackers from tampering with the predefined process. On requesting RA, a report with measurements based on the hash of the initial enclave state generated by the trusted processor is received. This facilitates the identification of the program and completes the memory layout. Intel EPID [132] signs this measurement, and the Intel Attestation Service verifies the correctness of the signature as a trusted third party. Consequently, verifiable and secure computations are performed in a remote enclave. Simultaneously, a secure key exchange is performed between

the enclave and the remote client within this RA protocol. Therefore, after performing RA, communication with a remote enclave can be initiated over a secure channel using AES-GCM and so on.

Fundamental Research. As mentioned above, TEE can process conventional encryption methods more flexibly and efficiently. For example, in [78], we propose PCT-TEE, an efficient and secure trajectory-based private contact tracing (PCT) system using TEE, which overcomes the weak points of existing Bluetooth-based PCT systems [135]. The use of TEE enables private contact tracing using trajectory data directly, thereby making it possible to detect *indirect contact*, which has been difficult in the existing approach. We formalize the trajectory-based private contact tracing problem as *spatio-temporal private set intersection* to be efficiently computed with TEE. The major challenge is how to design algorithms for a spatiotemporal private set intersection under the limited secure memory of the TEE. To this end, we design a TEE-based system with flexible trajectory data encoding algorithms. The experiments on real-world data show that the proposed system can process hundreds of PCT queries on tens of millions of records of trajectory data within a few seconds. The details of this work are presented in Appendix B.

On the other hand, while TEE has the attractive features described above, many studies have pointed out the vulnerabilities of TEE. In particular, memory/page access patterns or instruction traces can be exposed irrespective of the use of a TEE through side-channel attacks [84, 85, 136, 137, 86]. Note that the data itself is encrypted and cannot be viewed in enclaves, and an attacker can only see access patterns. This may lead to sensitive information being stolen from enclaves [136]. For example, cacheline-level access pattern leakage occurs when a malicious OS injects page faults [84] or uses page-table-based threats [85, 86]. Moreover, if a physical machine is accessible, probes may be attached to the memory bus directly. Designing effective and efficient defense methods against these attacks remains an important challenge in this context.

2.3.1 Federated Learning with TEE

Using TEE in FL is a promising approach [73, 74, 138, 72, 139]. In particular, TEE provides confidentiality, integrity, and functionalities such as Remote Attestation for verifiability, fully justifying its use on the untrusted server side in FL

[72, 76, 74]. After the first verification of the remote TEE, the shared gradients can be transmitted to the TEE via a secure channel and computed securely in a confidential, verifiable and arbitrary manner on the server side. PPFL [73] uses a TEE to hide parameters to prevent semi-honest client and server attacks on a global model. Citadel [72] addressed the important goal of making the design of models confidential in collaborative ML using TEE. However, side-channel attacks were not covered. In [74] and [138], the gradient aggregation step was taken to be hierarchical and/or partitioned using multiple servers such that the gradient information could only be partially observed by each server. Flatee [139] uses the combination of server-side TEE and DP in FL. These works did not provide any analysis and solution for the privacy leakages via memory access patterns that can be observable through side-channels. Olive [24] (introduced in Chapter 4) includes an analysis of access patterns in the aggregation procedure of FL and the design and demonstration of attack methods to motivate a proposed defense mechanism thoroughly in addition to specific solutions that lead to stronger security than any other method in FL on a single central TEE.

The major difference from centralized ML using TEE [140, 76] is that the training data are not shared with the server and are not centralized in the latter case, which can be critical because of privacy or contractual/regulatory reasons or for practical reasons, i.e., big and fast data at multiple edges. It is also important to outsource heavy computations required for ML training from TEE's limited computational resources to external clients.

CHAPTER 3

ULDP-FL: Federated Learning with Across Silo User-Level Differential Privacy

As shown in Table 1.2, a typical way to satisfy **B** and **C** is through DP-FL, which guarantees the DP of the trained model. Here, to apply DP-FL in practical scenarios, we point out that there is a critical gap in current DP-FL methodologies and study a method to improve it.

3.1 Introduction

Although DP is the de facto standard for privacy protection in ML, it has theoretical limitations. The standard DP definition [42] considers a single record as a unit of privacy. This can easily break down in realistic settings where one user may provide multiple records, potentially deteriorating the privacy loss bound of DP. To address this, the notion of *user-level DP* has been studied [80, 81, 82, 83]. In user-level DP, all records belonging to a single user are considered as a unit of privacy, which is a stricter definition than standard DP. We distinguish user-level DP from *group-privacy* [141], which considers any k records as privacy units. User-level DP has also been studied in the FL context [57, 58, 142, 143, 96] as

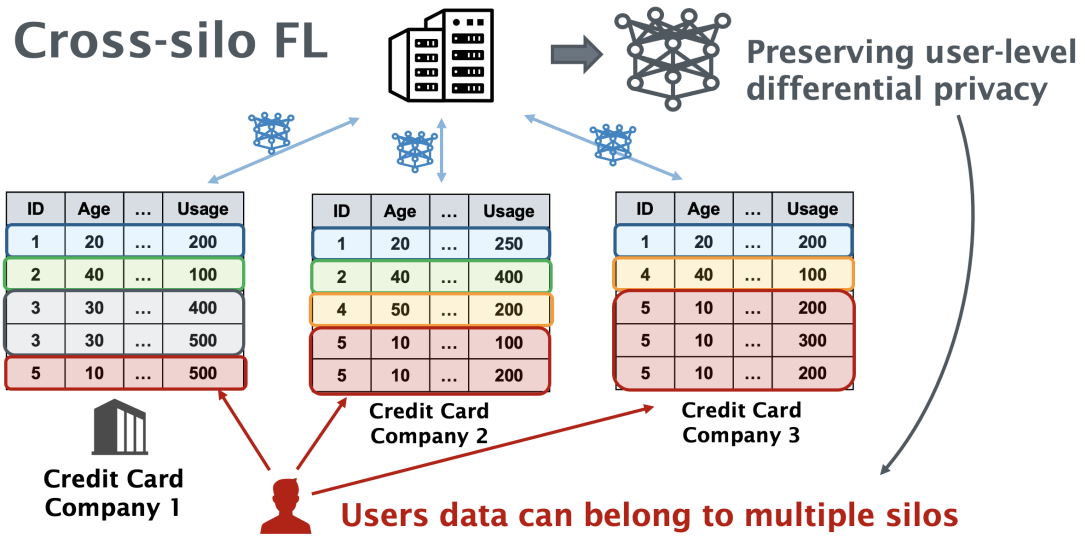


Figure 3.1: In cross-silo FL, in general, records belonging to the same user can exist across silos, e.g., a user can use several credit card companies. In this study, we investigate how to train models satisfying *user-level* DP in this setting.

client-level DP. However, these studies focus on the cross-device FL setting, where one user's data belongs to a single device only.

Cross-silo FL [106, 107, 144, 108] is a practical variant of FL in which a relatively small number of silos (e.g., hospitals or credit card companies) participate in training rounds. In cross-silo FL, unlike in cross-device FL, a single user can have multiple records across silos, as shown in Figure 3.1. Existing cross-silo DP-FL studies [107, 144, 108] have focused on record-level DP for each silo; user-level DP across silos has not been studied. Therefore, an important research question arises: *How do we design an FL framework that guarantees user-level DP across silos in cross-silo FL?*

A naive design for an algorithm that guarantees user-level DP combines bounding user contributions (number of records) as in [82, 83] and group-privacy property of DP [141]. Group-privacy simply extends the indistinguishability of record-level DP to multiple records. We can convert any DP algorithm to its group-privacy version of DP (Lemma 5, 6), which we formally define as Group DP (GDP) later in Definition 6. However, this approach can be impractical due to the super-linear privacy bound degradation of conversion to GDP (as shown in Figure 3.2) and the need to appropriately limit the maximum number of user

records (group size) in a distributed environment. In particular, the former issue is a fundamental limitation for DP and highlights the need to develop algorithms that directly satisfy user-level DP without requiring conversion to GDP.

In this study, we present a novel cross-silo and differentially private FL framework named ULDP-FL, designed to directly guarantee user-level DP through the incorporation of per-user weighted clipping. The contributions of this work are summarized as follows:

- We introduce a problem setting for cross-silo FL under user-level DP across silos, as illustrated in Figure 3.1.
- We propose the ULDP-FL framework and design baseline algorithms capable of achieving user-level DP across silos. The baseline algorithms combine limiting the maximum number of records per user and using group-privacy with DP-SGD [48] for each silo.
- Our algorithm ULDP-AVG/SGD directly satisfies user-level DP by implementing user-level weighted clipping within each silo, effectively bounding user-level sensitivity for an unlimited number of a single user's records across silos. We provide theoretical analysis on ULDP-AVG, showing a user-level DP bound and a convergence analysis.
- We evaluate the proposed method and baseline approaches through comprehensive experiments on various real-world datasets. The results underscore that our method yields superior trade-offs between privacy and utility compared to the baseline approaches.
- We further design an effective method by refining the weighting strategy for user-level clipping bounds. Since this approach may lead to additional privacy leakage of the training data, we develop a private protocol employing cryptographic techniques. We evaluate the extra computational overhead of the proposed private protocol using real-world benchmark scenarios.

3.2 Background & Preliminaries

3.2.1 Cross-silo Federated learning

In this work, we consider the following general cross-silo FL scenario. We have a central aggregation server and a set of silos S participating in all rounds. In cross-silo FL, we optimize the global model parameter cooperatively across all silos. Formally, the goal can be formulated for silo $s \in S$ as follows, as in Eq. (2.1),

$$\min_x \left\{ f(x) := \frac{1}{|S|} \sum_{s \in S} f_s(x) \right\}, \; f_s(x) := \mathbb{E}_{\xi \sim \mathcal{D}_s} F(x; \xi). \tag{3.1}$$

Additionally, we have a user set U across all datasets in silos, where each record belongs to one user $u \in U$, and each user may have multiple records in one silo and across multiple silos. Each silo s has local objectives for each user u, $f_{s,u} := \mathbb{E}_{\xi \sim \mathcal{D}_{s,u}} F(x; \xi)$, where $\mathcal{D}_{s,u}$ is the data distribution of s and u. In round $t \in [T]$ in FL, the global model parameter is denoted as x_t.

Note that this modeling is clearly different from cross-device FL in that there is no constraint that one user should belong to one device [58, 54]. Records from one user can belong to multiple silos. For example, the same customer may use several credit card companies. Additionally, all silos participate in all training rounds, unlike the probabilistic participation in cross-device FL [143], and the number of silos $|S|$ is small, around 2 to 100.

3.2.2 Differential Privacy for Multiple Records

Here are some notes on DP in this study. We label the original definition as *record-level* DP because the neighboring databases differ in only one record. In the privacy accounting of this study, we use Rényi DP (RDP) [50] because of the tightness of the analysis of the privacy composition. The definitions and lemmas of RDP are the original definitions for record-level neighboring databases (as in Definition 4), but are the same for user-level neighboring databases. We consider the $\ell 2$-norm ($\|\cdot\|_2$) as the $\ell 2$-sensitivity for following analysis due to adding Gaussian noise.

To extend privacy guarantees of DP to multiple records, group-privacy [141] has been explored as a solution. We refer to the group-privacy version of DP as

Group DP (GDP) and define it as follows:

Definition 6 ((k, ϵ, δ)-GDP)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (k, ϵ, δ)-GDP if, for any two input databases $D, D' \in \mathcal{D}$, s.t. D' differs from D in at most k records and any subset of outputs $Z \subseteq \mathcal{Z}$, Eq. (2.2) holds.*

GDP is a versatile privacy definition, as it can be applied to existing DP mechanisms without modification. To convert (ϵ, δ)-DP to (k, ϵ, δ)-GDP, it is known that any $(\epsilon, 0)$-DP mechanism satisfies $(k, k\epsilon, 0)$-GDP [141]. However, in the case of any $\delta > 0$, δ increases super-linearly [145], leading to a much larger ϵ (Lemma 5).

Lemma 5 (Group Privacy Conversion (Record-level DP to GDP) [145])**.** *If f is (ϵ, δ)-DP, for any two input databases $D, D' \in \mathcal{D}$ s.t. D' differs from D in at most k records and any subset of outputs $Z \subseteq \mathcal{Z}$, it holds that*

$$\Pr[f(D) \in Z] \leq \exp(k\epsilon) \Pr[f(D') \in Z] + ke^{(k-1)\epsilon}\delta.$$

It means when f is (ϵ, δ)-DP, f satisfies $(k, k\epsilon, k \exp^{(k-1)\epsilon} \delta)$-GDP.

Also, we can compute GDP using group-privacy property of Rényi DP [50]. First, we calculate the RDP of the algorithm, then convert it to group version of RDP, and subsequently to GDP (Lemma 6).

Lemma 6 (Group-Privacy of RDP (Record-level DP to GDP) [50])**.** *If $f : D \to \mathbb{R}^d$ is (α, ρ)-RDP, $g : D' \to D$ is k-stable and $\alpha \geq 2^{k+1}$, then $f \circ g$ is $(\alpha/2^k, 3^k\rho)$-RDP.*

Here, group-privacy property is defined using a notion of k-stable transformation [110]. $g : D' \to D$ is k-stable if $g(A)$ and $g(B)$ are neighboring in D implies that there exists a sequence of length $c + 1$ so that $D_0 = A, ..., D_c = B$ and all (D_i, D_{i+1}) are neighboring in D'. This privacy notion corresponds to $(k, \cdot, \cdot)$-GDP in Definition 6.

Here, to highlight the significant privacy degradation of GDP, we conduct a pre-experiment and show the converted privacy bounds for increasing group sizes. Figure 3.2 illustrates a numerical comparison of the group-privacy conversion from DP to GDP with normal DP (Lemma 5) and RDP (Lemma 6). To compute the final GDP privacy bounds, we repeatedly run the Gaussian mechanism with $\sigma = 5.0$ and a sampling rate of 0.01 for 10^5 iterations, emulating a typical DP-SGD execution setup. We use a fixed $\delta = 10^{-5}$ and vary the group size

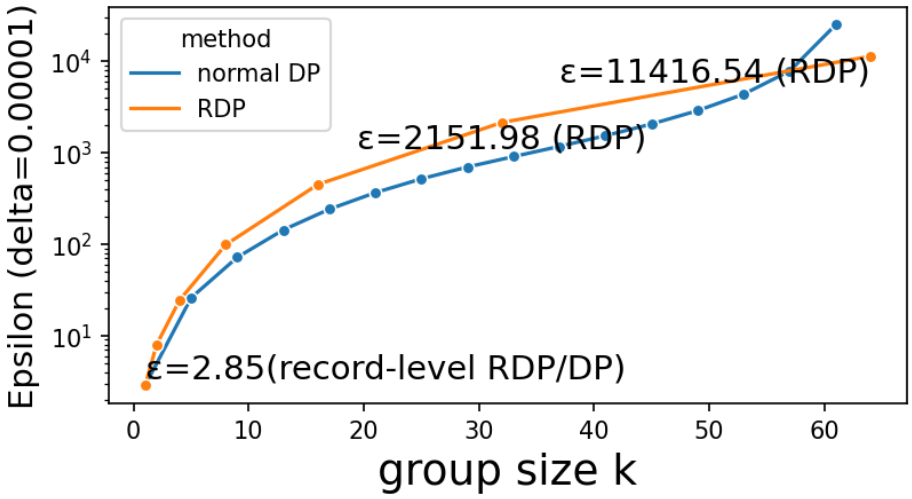


Figure 3.2: Group-privacy conversion results.

$k = 1, 2, 4, 8, 16, 32, 64$. To compute GDP, RDP for sub-sampled Gaussian mechanisms is calculated according to [49]. We then compute GDP using group-privacy of RDP by Lemma 6. For normal DP, the computed RDP is converted to normal DP by Lemma 2, and then to GDP by Lemma 5.

When converting from normal DP to GDP, computing the final ϵ at a fixed δ is challenging. In Lemma 2, the output ϵ (denoted as ϵ_2) depends on the input δ (denoted as δ_2), and the final δ (denoted as δ_5) output by Lemma 5 depends on both ϵ_2 and δ_2. Therefore, we repeatedly select δ_2 in a binary search manner, compute ϵ_2 and δ_2, and finally report the ϵ when the difference between δ_2 and $\delta = 10^{-5}$ is sufficiently small (accuracy by 10^{-8}) as the ϵ of GDP [a]. Note that this method does not guarantee achieving the optimal ϵ for the given δ, but it finds a reasonable ϵ.

In the figure, we plot various group sizes, k, on the x-axis and ϵ of GDP at a fixed $\delta = 10^{-5}$ on the y-axis. Significantly, the results indicate that as the group size k increases, ϵ grows rapidly, highlighting a considerable degradation in the privacy bound of GDP. For instance, with $\epsilon = 2.85$ at record-level ($k = 1$), the value reaches 2100 for only $k = 32$, and 11400 at $k = 64$. While there may be some looseness in the group-privacy conversion of RDP compared to normal DP for some small group sizes, the difference is relatively minor (roughly three times

[a] The implementation is in the function `get_normal_group_privacy_spent()` in `https://github.com/FumiyukiKato/uldp-fl/blob/main/src/noise_utils.py`

at most). Moreover, the RDP conversion is easier to compute with a fixed δ. Hence, we utilize RDP conversion in our experiments.

3.2.3 DP-FL in Cross-silo Setting

DP has been applied to the FL paradigm, with the goal of ensuring that the trained model satisfies DP. A popular DP variant in the context of cross-device FL is user-level DP (also known as client-level DP) [58, 146, 96]. Informally, this definition ensures indistinguishability for device participation and has demonstrated a favorable privacy-utility trade-off even with large-scale models [143]. These studies often employ secure aggregation [66, 147] to mitigate the need for trust in other parties during FL model training. This is achieved by allowing the server and other silos to only access appropriately perturbed models after aggregation, often referred to as Distributed DP [96, 148]. In particular, shuffling-based variants have recently gained attention [148, 61, 60] and are being deployed in FL [123], which also provides user-level DP. All of these studies assume that a single device holds all records for a single user, i.e., cross-device FL. However, in a cross-silo setting, this definition does not extend meaningful privacy protection to individual users when they possess multiple records across silos.

Another DP definition in cross-silo FL offers record-level DP within each silo [107, 144, 108], referred to as Silo-specific sample-level or Inter-silo record-level DP. These studies suggest that record-level DP can guarantee user-level DP through group-privacy [141]. However, they cannot account for settings where a single user may have records across multiple silos. To the best of our knowledge, no method exists for training models that satisfy user-level DP in cross-silo FL where a single user's records may extend across multiple silos.

3.3 ULDP-FL Framework

3.3.1 Trust Model and Assumptions

We assume that all (two or more) silos and aggregation servers are *semi-honest*, meaning they observe the information but do not deviate from the protocol. This is a typical assumption in prior works [66, 149]. In our study, aggregation is performed using secure aggregation to ensure that the server only gains access

to the model after aggregation [96]. All communications between the server and silos are encrypted with SSL/TLS, and third parties with the ability to snoop on communications cannot access any information except for the final trained model. We assume that there is no collusion, which is reasonable given that silos are socially separate institutions (such as different hospitals or companies). Additionally, in our scenario, we assume that record linkage [150] across silos has already been completed, resulting in shared common user IDs. Both the server and the silos are aware of the total number of users $|U|$ with at least one record and the number of silos $|S|$. When sub-sampling is employed for DP amplification, only the server is permitted to know the sub-sampling results for each round [96, 146]. It is important to note that all these assumptions do not affect the privacy guarantees for external users.

3.3.2 Privacy Definition

In contrast to GDP, which offers indistinguishability for any k records, user-level DP [81, 58] provides a more reasonable user-level indistinguishability. While [58] focuses solely on a cross-device FL context, we re-establish user-level DP (ULDP) in the cross-silo setting as follows:

Definition 7 ((ϵ, δ)-ULDP)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{Z}$ satisfies (ϵ, δ)-ULDP if, for any two input databases across silos $D, D' \in \mathcal{D}$, s.t. D' differs from D in at most one user's records, and any $Z \subseteq \mathcal{Z}$, Eq. (2.2) holds.*

The fundamental difference from record-level DP lies in the definition of the neighboring databases, which inherently defines user-level sensitivity. Additionally, it is important to emphasize that the input database D represents the comprehensive database spanning across silos.

If the number of records per user in the database is less than or equal to k, it is clear that GDP is a generalization of ULDP, and the following proposition holds.

Proposition 1. *If a randomized mechanism $\mathcal{M}$ is (k, ϵ, δ)-GDP with input database D in which any user has at most k records, the mechanism $\mathcal{M}$ with input database D also satisfies (ϵ, δ)-ULDP.*

One drawback of GDP is the challenge of determining the appropriate value for k. Setting k to the maximum number of records associated with any individual

user could lead to introducing excessive noise to achieve the desired privacy protection level. On the other hand, if a smaller k is chosen, the data of users with more than k records must be excluded from the dataset, potentially introducing bias and compromising model utility. In this context, while several studies have analyzed the theoretical utility for a given k [80, 81] and theoretical considerations for determining k have been partially explored in [83], it still remains an open problem. In contrast, ULDP does not necessitate the determination of k. Instead, it requires designing a specific ULDP algorithm.

3.3.3 Baseline Methods: ULDP-NAIVE/GROUP-k

We begin by describing two baseline methods. The first method is ULDP-NAIVE (described in Algorithm 3), a straightforward approach using substantial noise. It extends DP-FedAVG [143], where each silo locally optimizes with multiple epochs, computes the model delta, clips by C, and adds Gaussian noise with variance σ^2C^2. However, in ULDP-NAIVE, since a single user may contribute to the model delta of all silos, the sensitivity across silos is $|C| * S$ for the aggregated model delta (Line 15). Compared to DP-FedAVG, which focuses on cross-device FL, the number of model delta samples (number of silos as opposed to the number of devices) is very small, resulting in larger variance. Thus, ULDP-NAIVE satisfies ULDP but at a significant sacrifice in utility. The aggregation is performed using secure aggregation and is assumed to be so in the following algorithms.

Theorem 3. *For any $0 < \delta < 1$ and $\alpha > 1$, given noise multiplier σ, ULDP-NAIVE satisfies $(\epsilon = \frac{T\alpha}{2\sigma^2} + \log\left((\alpha-1)/\alpha\right) - (\log\delta + \log\alpha)/(\alpha-1), \delta)$-ULDP after T rounds.* (The actual value of ϵ is numerically calculated by selecting the optimal α so that ϵ is minimized.)

Proof. At each round t, due to the clipping operation (Line 14 in Algorithm 3), for each silo $s \in S$, any user's contribution to Δ_t^s (Line 6) is limited to at most C (regardless of the number of user records). Since a single user may exist in any silo, they contribute at most $|S|C$ to $\sum_{s\in S}\Delta_t^s$, which means the user-level sensitivity is $|S|C$. Therefore, when each silo adds Gaussian noise with variance $\sigma^2C^2|S|$, the aggregate $\sum_{s\in S}\Delta_t^s$ includes Gaussian noise with variance $\sigma^2C^2|S|^2$. Then, by Lemma 3, it satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP for $\alpha > 1$. And after T rounds, it satisfies $(\alpha, \frac{T\alpha}{2\sigma^2})$-RDP by Lemma 1. Finally, by Lemma 2, we obtain the final result. □

Algorithm 3 ULDP-NAIVE

Input: S: silo set (silo $s \in S$), η_l: local learning rate, η_g: global learning rate, σ: noise parameter, C: clipping bound, T: total round, Q: #local epochs

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, C, \sigma, \eta_l)$
6: **end for**
7: $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{|S|} \sum_{s \in S} \Delta_t^s$
8: **end for**
9: **end procedure**
10: **procedure** CLIENT(x_t, C, σ, η_l)
11: $x_s \leftarrow x_t$
12: **for** epoch $q = 0, 1, \ldots, Q-1$ **do**
13: Compute stochastic gradients $g_{t,q}^{(s)}$
14: ▷ $\mathbb{E}[g_{t,q}^{(s)}] = \nabla f_s(x_s)$
15: $x_s \leftarrow x_s - \eta_l g_{t,q}^{(s)}$
16: **end for**
17: $\Delta_t \leftarrow x_t - x_s$
18: $\tilde{\Delta}_t \leftarrow \Delta_t \cdot \min\left(1, \frac{C}{\|\Delta_t\|_2}\right)$ ▷ clipping with C
19: $\Delta_t' \leftarrow \tilde{\Delta}_t + \mathcal{N}(0, I\sigma^2 C^2 |S|)$ ▷ huge noise based on user-level sensitivity
20: **return** Δ_t'
21: **end procedure**

Secondly, we introduce the baseline algorithm ULDP-GROUP-k (described in Algorithm 4), which combines the constraint of limiting each user's records to a given k while satisfying (k, ϵ, δ)-GDP. As Proposition 1 implies, this ensures (ϵ, δ)-ULDP. The algorithm achieves GDP by implementing DP-SGD [48] within each silo. The core principle of the algorithm is similar to that of [107], except for the global setting of a single privacy budget across silos. Before executing DP-SGD, we constrain the number of records per user to k (Line 8). This is accomplished by employing flags, denoted as $\mathbf{B}$, which indicate the records to be used for training (i.e., $b^s u, i = 1$), with a total of k records for each user across all silos (i.e., $\forall u, \sum_{s,i} b_{u,i}^s \leq k$). These flags must be consistent across all rounds to ensure

Algorithm 4 ULDP-GROUP-k

Input: S: silo set (silo $s \in S$), η_l: local learning rate, η_g: global learning rate, σ: noise parameter, D_s: training dataset of silo s, C: clipping bound, T: total round, Q: #local epochs, k: group size, γ: sampling rate, $\mathbf{B}$: flags for limit contribution s.t. for each matrix $\mathbf{b}^s \in \mathbf{B}$ if $b^s_{u,i} = 1$ the user u's i-th record in silo s is used in training dataset, otherwise the record is excluded

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta^s_t \leftarrow$ CLIENT$(x_t, C, \sigma, \eta_l, \gamma, \mathbf{b}^s)$
6: **end for**
7: $\theta_{t+1} \leftarrow \theta_t + \eta_g \frac{1}{|S|} \sum_{s \in S} \Delta^s_t$
8: **end for**
9: **end procedure**
10: **procedure** CLIENT$(x_t, C, \sigma, \eta_l, \gamma, \mathbf{b}^s)$
11: $D'_s \leftarrow$ filter D_s by $\mathbf{b}^s$
12: $x^Q_t \leftarrow$DP-SGD$(\theta_t, D'_s, C, \sigma, \eta_l, \gamma, Q)$
13: ▷ Algorithm 1 in [48]
14: $\Delta_{t+1} \leftarrow x^Q_t - x_t$
15: **return** Δ_{t+1}
16: **end procedure**

privacy. We disregard the privacy concerns in generating these flags as this is a baseline method. We then perform DP-SGD to satisfy record-level DP (Line 9), which is subsequently converted to GDP.

Theorem 4. *For any $0 < \delta < 1$, any integer k to the power of 2 and $\alpha > 2^{k+1}$, ULDP-GROUP-k satisfies $(3^k\rho + \log((\frac{\alpha}{2^k} - 1)/\frac{\alpha}{2^k}) - (\log\delta + \log\frac{\alpha}{2^k})/(\frac{\alpha}{2^k} - 1), \delta)$-ULDP where $\rho = \max_{s \in S} \rho_s$ s.t. for each silo $s \in S$, DP-SGD of local subroutine satisfies (α, ρ_s)-RDP.*

Proof. In each silo $s \in S$, by performing DP-SGD with Q epochs and T rounds, we achieve record-level $(\alpha, \rho_s = \rho(\sigma, \gamma, QT))$-RDP. The actual value of ρ_s is calculated numerically as described in [51]. RDP is known to satisfy parallel composition [151]. That is, for disjoint databases D_1 and D_2, their combined release

$(\mathcal{M}_1(D_1), \mathcal{M}_2(D_2))$ where $\mathcal{M}_1$ is (α, ρ_1)-RDP and $\mathcal{M}_2$ is (α, ρ_2)-RDP satisfies $(\alpha, \max\{\rho_1, \rho_2\})$-RDP. Since input databases D_s for any silo $s \in S$ are disjoint from others, after T rounds, the trained model satisfies $(\alpha, \rho = \max_{s \in S} \rho_s)$-RDP for the entire cross-silo database $D = D_1 \oplus ... \oplus D_{|S|}$. Then, by applying Lemma 6 and Lemma 2, for any integer k that is a power of 2 and any $\alpha \geq 2^{k+1}$, it also satisfies $(k, 3^k\rho + \log\left((\frac{\alpha}{2^k} - 1)/\frac{\alpha}{2^k}\right) - (\log\delta + \log\frac{\alpha}{2^k})/(\frac{\alpha}{2^k} - 1), \delta)$-GDP. By filtering with $\mathbf{B}$, we ensure that any user has at most k records in D. The final result is obtained by Proposition 1. (An alternative method to compute ϵ is to use Lemma 5 instead of Lemma 6.) □

While ULDP-GROUP shares algorithmic similarities with existing record-level DP cross-silo FL frameworks [107], it presents weaknesses from several perspectives: (1) Significant degradation of privacy bounds due to the group-privacy conversion (DP to GDP). (2) The challenge of determining an appropriate group size k [83], which requires substantial insights into data distribution across silos and might breach the trust model. The determination of the flags $\mathbf{B}$ can also be problematic. (3) The use of group-privacy to guarantee ULDP necessitates removing records from the training dataset, potentially introducing bias and causing utility degradation [83, 152]. Our next proposed method aims to address these challenges.

3.3.4 Advanced methods: ULDP-AVG/SGD

To directly satisfy ULDP without using group-privacy, we designed ULDP-AVG (Algorithm 5) and ULDP-SGD (Algorithm 6). These can be seen as variants of DP-FedAVG and DP-FedSGD [143]. In most cases, DP-FedAVG is preferred in terms of privacy-utility trade-off and communication cost, while DP-FedSGD might be preferable only when we have fast networks [143], which is also the case for ULDP-AVG and ULDP-SGD. In the following analysis, we focus on ULDP-AVG.

Intuitively, ULDP-AVG limits each user's contribution to the model by training the model for each user in each silo and performing per-user per-silo clipping across all silos with globally prepared clipping weights. In each round, ULDP-AVG computes parameter deltas using a per-user dataset in each silo to achieve ULDP: selecting a user (Line 8), training local model with Q epochs using only the selected user's data (Lines 10-13), calculating model delta (Line 14) and clipping

Algorithm 5 ULDP-AVG

Input: U: user set (user $u \in U$), S: silo set (silo $s \in S$), η_l: local learning rate, η_g: global learning rate, σ: noise parameter, C: clipping bound, T: total round, Q: #local epochs, $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_{|S|})$: matrix with weight for user u and silo s, and $\forall u \in U$, $w_{s,u} \in \mathbf{w}_s$ and $\sum_{s \in S} w_{s,u} = 1$

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $\Delta_t^s \leftarrow \text{CLIENT}(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$
6: **end for**
7: $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{|U||S|} \sum_{s \in S} \Delta_t^s$
8: **end for**
9: **end procedure**
10: **procedure** CLIENT$(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$
11: **for** user $u \in U$ **do** ▷ per-user training with $\mathcal{D}_{s,u}$
12: $x_t^{s,u} \leftarrow x_t$
13: **for** epoch $q = 0, 1, \ldots, Q-1$ **do**
14: Compute stochastic gradients $g_{t,q}^{s,u}$
15: ▷ $\mathbb{E}[g_{t,q}^{s,u}] = \nabla f_{s,u}(x_t^{s,u})$
16: $x_t^{s,u} \leftarrow x_t^{s,u} - \eta_l g_{t,q}^{s,u}$
17: **end for**
18: $\Delta_t^{s,u} \leftarrow x_t^{s,u} - x_t$
19: $\tilde{\Delta}_t^{s,u} \leftarrow w_{s,u} \cdot \Delta_t^{s,u} \cdot \min\left(1, \frac{C}{\|\Delta_t^{s,u}\|_2}\right)$ ▷ per-user weighted clipping
20: **end for**
21: $\Delta_t^s \leftarrow \sum_{u \in U} \tilde{\Delta}_t^{s,u} + \mathcal{N}(0, I\sigma^2 C^2 / |S|)$
22: **return** Δ_t^s
23: **end procedure**

the delta (Line 15). These clipped deltas $\Delta_t^{s,u}$ are then weighted by $w_{s,u}$ (Line 15) and summed for all users (Line 16). As long as the weights $w_{s,u}$ satisfy constraints $\forall u \in U$, $w_{s,u} > 0$ and $\sum_{s \in S} w_{s,u} = 1$, each user's contribution, or *sensitivity*, to the delta aggregation $\sum_{s \in S} \Delta_t^s$ is limited to C at most. This allows ULDP-AVG to provide user-level privacy. We will discuss better ways to determine $\mathbf{W}$ later, but a simple way is to set $w_{s,u} = 1/|S|$. Compared to DP-FedAVG, ULDP-AVG

Algorithm 6 ULDP-SGD

Input: U: user set (user $u \in U$), S: silo set (silo $s \in S$), η_g: global learning rate, σ: noise parameter, C: clipping bound, T: total round, $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_{|S|})$: matrix with weight for user u and silo s, and $\forall u \in U$, $w_{s,u} \in \mathbf{w}_s$ and $\sum_{s \in S} w_s^{(u)} = 1$

1: **procedure** SERVER
2: Initialize model x_0
3: **for** each round $t = 0, 1, \ldots, T-1$ **do**
4: **for** each silo $s \in S$ **do**
5: $g_t^s \leftarrow \text{CLIENT}(x_t, \mathbf{w}_s, C, \sigma)$
6: **end for**
7: $x_{t+1} \leftarrow x_t - \eta_g \frac{1}{|U||S|} \sum_{s \in S} g_t^s$
8: **end for**
9: **end procedure**
10: **procedure** CLIENT$(x_t, \mathbf{w}_s, C, \sigma)$
11: **for** user $u \in U$ **do** ▷ per-user training
12: Compute stochastic gradients $g_t^{s,u}$
13: ▷ $\mathbb{E}[g_t^{s,u}] = \nabla f_{s,u}(x_s)$
14: $\tilde{g}_t^{s,u} \leftarrow w_{s,u} \cdot g_t^{s,u} \cdot \min\left(1, \frac{C}{\|g_t^{s,u}\|_2}\right)$
15: **end for**
16: $g_t^s \leftarrow \sum_{u \in U} \tilde{g}_t^{s,u} + \mathcal{N}(0, I\sigma^2 C^2 / |S|)$
17: **return** g_t^s
18: **end procedure**

increases computational cost due to per-user local training iteration but keeps communication costs the same, which is likely acceptable in the cross-silo FL setting.

Theorem 5. *For any $0 < \delta < 1$ and $\alpha > 1$, given noise multiplier σ, ULDP-AVG satisfies $(\epsilon = \frac{T\alpha}{2\sigma^2} + \log((\alpha-1)/\alpha) - (\log\delta + \log\alpha)/(\alpha-1), \delta)$-ULDP after T rounds.*

Proof. For any round t, due to the clipping operation (Line 15), for each silo $s \in S$ any user's contribution to Δ_t^s (Line 6) is limited to at most $w_{s,u}C$ (regardless of the number of user records). Therefore, for $\sum_{s \in S} \Delta_t^s$ (Line 6), any user's contribution is at most $\sum_{s \in S} w_{s,u} C = C$ since $\sum_{s \in S} w_{s,u} = 1$. That means

Algorithm 7 ULDP-AVG with user-level sub-sampling

Input: U: user set (user $u \in U$), S: silo set (silo $s \in S$), η_l: local learning rate, η_g: global learning rate, σ: noise parameter, C: clipping bound, T: total round, Q: #local epochs, $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_{|S|})$: matrix with weight for user u and silo s, and $\forall u \in U$, $w_{s,u} \in \mathbf{w}_s$ and $\sum_{s \in S} w_{s,u} = 1$, q: user-level sub-sampling probability

procedure SERVER
  Initialize model x_0
  for each round $t = 0, 1, \ldots, T-1$ **do**
    $U_t \leftarrow$ Poisson sampling from U with probability q
    for each silo $u \in U_t$ **do**
      for each silo $s \in S$ **do**
        $w_{s,u} \leftarrow 0$
      end for
    end for
    for each silo $s \in S$ **do**
      $\Delta_t^s \leftarrow$ CLIENT$(x_t, \mathbf{w}_s, C, \sigma, \eta_l)$ ▷ CLIENT is the same as Algorithm 5
    end for
    $x_{t+1} \leftarrow x_t + \eta_g \frac{1}{q|U||S|} \sum_{s \in S} \Delta_t^s$
  end for
end procedure

the user-level sensitivity is just C. Since each silo adds Gaussian noise with variance $\sigma^2 C^2 / |S|$, $\sum_{s \in S} \Delta_t^s$ includes Gaussian noise with variance $\sigma^2 C^2$. Then, by Lemma 3, it satisfies $(\alpha, \frac{\alpha}{2\sigma^2})$-RDP for $\alpha > 1$ in a user-level manner. And after T rounds, it satisfies $(\alpha, \frac{T\alpha}{2\sigma^2})$-RDP by Lemma 1. Finally, by Lemma 2, we obtain the result. □

Remark 1. For further privacy amplification, we introduce user-level sub-sampling, which can make RDP smaller according to the sub-sampled amplification theorem (Lemma 4) [49]. User-level sub-sampling must be done globally across silos. This sub-sampling can be implemented in the central server by controlling the weight $\mathbf{W}$ for each round, i.e., all users not sub-sampled are set to 0 as shown in Algorithm 7. This may violate privacy against the server but does not affect the DP when the final model is provided externally as discussed in C.3 of [146]. We have detailed algorithms and experimental results to show the effectiveness of user-level sub-sampling in our experiments.

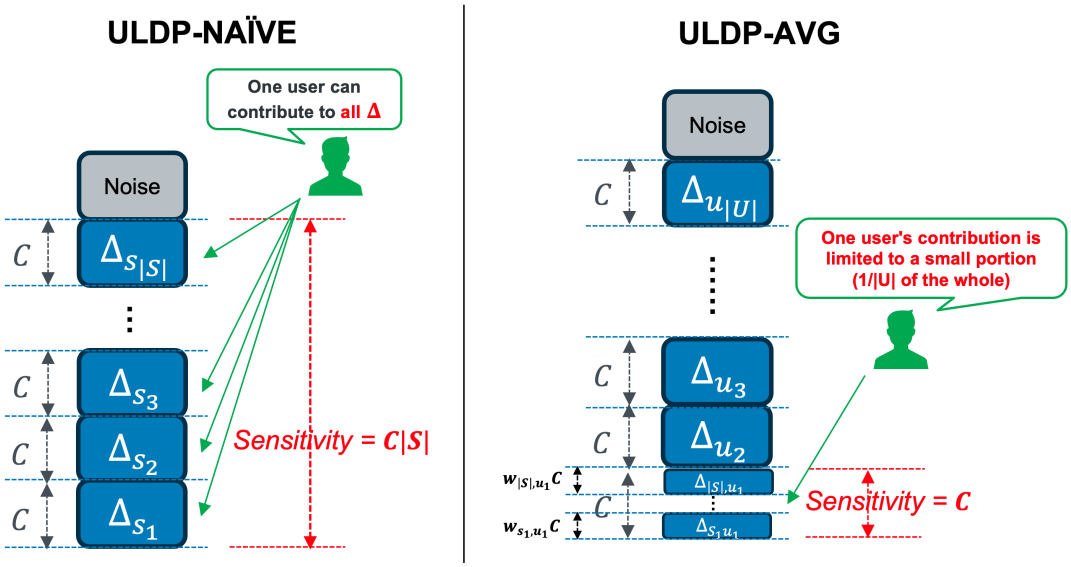


Figure 3.3: An intuitive illustration of the difference between ULDP-NAIVE and ULDP-AVG. In ULDP-NAIVE, every single user can contribute to all model deltas. On the other hand, in ULDP-AVG, one user's contribution is limited to a small portion, i.e., $1/|U|$ of the whole model delta. This reduces sensitivity. When $|U|$ is large, which often happens in cross-silo FL, it can be a particular advantage.

Comparison to Baselines. Compared to ULDP-GROUP, ULDP-AVG satisfies ULDP without group-privacy, thus avoiding the large privacy bound caused by group-privacy conversion, the need to choose a group size k, and removing records. ULDP-AVG can be used for an arbitrary number of records per user. Also, we illustrate the intuitive difference between ULDP-NAIVE and ULDP-AVG in Figure 3.3. Fundamentally, per-user clipping can be viewed as cross-user FL (instead of cross-silo FL), ensuring that each user contributes only to their user-specific portion of the aggregated model updates (i.e., $\sum_{s\in S} \tilde{\Delta}_t^{s,u}$) instead of the entire aggregated update (i.e., $\sum_{s\in S} \Delta_t^s$), thereby reducing sensitivity (as illustrated in Figure 3.3). The user contributes only $1/|U|$ of the entire aggregated model update, which is especially effective when $|U|$ is large, as in cross-silo FL (i.e., $|S| \ll |U|$). Moreover, computing the model delta at the user level leads to lower Gaussian noise variances due to large $|U|$, while it also introduces new biases. The overhead due to such biases can also be seen in the convergence analysis, motivating a better weighting strategy to reduce this overhead.

Convergence Analysis

Here, we theoretically analyze our proposed algorithm and provide a convergence analysis to compare it with existing methods. To this end, we use the following assumptions. Each of these is a standard assumption in non-convex optimization. In particular, the second assumption, σ_g, quantifies the heterogeneity of non-i.i.d. data between silos in FL and is used in many previous studies. $\sigma_g = 0$ corresponds to the i.i.d. setting. For each $s \in S$ and $u \in U$, we assume access to an unbiased stochastic gradient $g_{t,q}^{s,u}$ of the true local gradient $\nabla f_{s,u}(x)$ for s and u.

Assumption 1 (Lipschitz Gradient)**.** *The function $f_{s,u}$ is L-smooth for all silo $s \in S$ and user $u \in U$, i.e., $\|\nabla f_{s,u}(x) - \nabla f_{s,u}(y)\| \leq L\|x - y\|$, for all $x, y \in \mathbb{R}^d$.*

Assumption 2 (Bounded Local and Global Variance)**.** *The function $f_{s,u}$ is σ_l-locally-bounded, i.e., the variance of each local gradient estimator is bounded as $\mathbb{E}[\|g_{t,q}^{s,u} - \nabla f_{s,u}(x_{t,q}^{s,u})\|^2] \leq \sigma_l^2$ for all s, u and t, q. And the functions are σ_g-globally-bounded, i.e., the global gradient variance is bounded as $\|\nabla f_{s,u}(x_t) - f(x_t)\|^2 \leq \sigma_g^2$ for all s, u, t.*

Assumption 3 (Globally Bounded Gradients)**.** *For all s, u, t, q, gradient is G-bounded, i.e., $\|g_{t,q}^{s,u}\| \leq G$.*

Theorem 6 (Convergence analysis on ULDP-AVG)**.** *For ULDP-AVG, with assumptions 1, 2 and 3 and $\min_x f(x) \geq f^*$, let local and global learning rates η_l, η_g be chosen as s.t. $\eta_g \eta_l \leq \frac{1}{3QL\bar{\alpha}_t}$ and $\eta_l < \frac{1}{\sqrt{30}QL}$, we have,*

$$
\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \\
\leq\ & \frac{1}{cT\eta_g\eta_l Q|S|}\left(\mathbb{E}\left[\frac{f(x_0)}{\underline{C}}\right] - \mathbb{E}\left[\frac{f^*}{\bar{C}}\right]\right) \\
&+\frac{5}{2c}L^2 Q\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + \frac{3\bar{C}L\eta_g\eta_l\sigma_l^2}{2c|S|^2|U|} + \frac{L\eta_g\sigma^2 C^2 d}{2c\underline{C}\eta_l Q|S||U|^2} \\
&+A_1\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}(|\alpha_t^{s,u} - \tilde{\alpha}_t^{s,u}| + |\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t|)\right] \\
&+A_2\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}\left(|\alpha_t^{s,u} - \tilde{\alpha}_t^{s,u}|^2 + |\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t|^2\right)\right] \qquad (3.2)
\end{aligned}
$$

where $c > 0$, $\bar{C} := \max_{s,u,t}\left(\frac{C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]} g_{t,q}^{s,u}\right]\|\right)}\right)$, $\underline{C} := \min_{s,u,t}\left(\frac{C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]} g_{t,q}^{s,u}\right]\|\right)}\right)$, $\alpha_t^{s,u} := \frac{w_{s,u}C}{\max\left(C,\eta_l\|\sum_{q\in[Q]} g_{t,q}^{s,u}\|\right)}$, $\tilde{\alpha}_t^{s,u} := \frac{w_{s,u}C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]} g_{t,q}^{s,u}\right]\|\right)}$, $\bar{\alpha}_t := \frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\tilde{\alpha}_t^{s,u}$, $A_1 := \frac{G^2}{c\underline{C}|U|T}$ *and* $A_2 := \frac{3L\eta_g\eta_l QG^2}{2c\underline{C}|U|T}$.

Proof. The many of techniques in the following proof are seen in [89, 153, 154].

For convenience, we define following quantities:

$$\alpha_t^{s,u} := \frac{w_{s,u}C}{\max\left(C,\eta_l\|\sum_{q\in[Q]} g_{t,q}^{s,u}\|\right)},\ \tilde{\alpha}_t^{s,u} := \frac{w_{s,u}C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]} g_{t,q}^{s,u}\right]\|\right)},\ \bar{\alpha}_t := \frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\tilde{\alpha}_t^{s,u},$$

$$\Delta_t^{s,u} := -\eta_l\sum_{q\in[Q]} g_{t,q}^{s,u}\cdot\alpha_t^{s,u},\ \tilde{\Delta}_t^{s,u} := -\eta_l\sum_{q\in[Q]} g_{t,q}^{s,u}\cdot\tilde{\alpha}_t^{s,u},\ \bar{\Delta}_t^{s,u} := -\eta_l\sum_{q\in[Q]} g_{t,q}^{s,u}\cdot\bar{\alpha}_t, \tag{3.3}$$

$$\breve{\Delta}_t^{s,u} := -\eta_l\sum_{q\in[Q]}\nabla f_{s,u}(x_{t,q}^{s,u})\cdot\bar{\alpha}_t,\ \Delta_t^s := \sum_{u\in U}\Delta_t^{s,u},\ \tilde{\Delta}_t^s := \sum_{u\in U}\tilde{\Delta}_t^{s,u},\ \bar{\Delta}_t^s := \sum_{u\in U}\bar{\Delta}_t^{s,u},\ \breve{\Delta}_t^s := \sum_{u\in U}\breve{\Delta}_t^{s,u}. \tag{3.4}$$

where expectation in $\tilde{\alpha}_t^{s,u}$ is taken over all possible randomness. Due to the smoothness in Assumption 1, we have

$$f(x_{x+1}) \le f(x_t) + \langle\nabla f(x_t), x_{t+1} - x_t\rangle + \frac{L}{2}\|x_{t+1} - x_t\|^2. \tag{3.5}$$

The model difference between two consecutive iterations can be represented as

$$x_{x+1} - x_t = \eta_g\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s + z_t^s) \tag{3.6}$$

with random noise $z_t^s \sim \mathcal{N}(0, I\sigma^2C^2/|S|)$. Taking expectation of $f(x_{t+1})$ over the randomness at communication round t, we have:

$$\begin{aligned}
\mathbb{E}\left[f(x_{x+1})\right] &\le f(x_t) + \langle\nabla f(x_t), \mathbb{E}\left[x_{t+1} - x_t\right]\rangle + \frac{L}{2}\mathbb{E}\left[\|x_{t+1} - x_t\|^2\right] \\
&= f(x_t) + \eta_g\langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s + z_t^s)\right]\rangle + \frac{L}{2}\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s + z_t^s)\right\|^2\right] \\
&= f(x_t) + \eta_g\langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s\right]\rangle + \frac{L}{2}\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s\right\|^2\right] + \frac{L}{2}\eta_g^2\frac{1}{|S|^2|U|^2}\sigma^2C^2d
\end{aligned} \tag{3.7}$$

where d in the last expression is dimension of x_t and in the last equation we use the fact that z_t^s is zero mean normal distribution.

Firstly, we evaluate the first-order term of Eq. (3.7),

$$\begin{aligned}
&\langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s\right]\rangle \\
&= \langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s - \tilde{\Delta}_t^s)\right]\rangle + \langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\tilde{\Delta}_t^s - \bar{\Delta}_t^s)\right]\rangle + \langle\nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\bar{\Delta}_t^s\right]\rangle,
\end{aligned} \tag{3.8}$$

and the last term of Eq. (3.8) can be evaluated as follows:

$$\begin{aligned}&\langle \nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\bar{\Delta}_t^s\right]\rangle \\ &= -\frac{\eta_l\bar{\alpha}_t Q}{2}\|\nabla f(x_t)\|^2 - \frac{\eta_l\bar{\alpha}_t}{2Q}\mathbb{E}\left[\left\|\frac{1}{\eta_l|S||U|\bar{\alpha}_t}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] + \frac{\eta_l\bar{\alpha}_t}{2}\mathbb{E}\left[\left\|\sqrt{Q}\nabla f(x_t) + \frac{1}{\sqrt{Q}\eta_l|S||U|\bar{\alpha}_t}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right].\end{aligned} \tag{3.9}$$

This is because $\mathbb{E}\bar{\Delta}_t^s = \breve{\Delta}_t^s$ and $\langle a, b\rangle = -\frac{1}{2}\|a\|^2 - \frac{1}{2}\|b\|^2 + \frac{1}{2}\|a+b\|^2$ for any vector a, b.

We further upper bound the last term of Eq. 3.9 as:

$$\begin{aligned}\mathbb{E}\left[\left\|\sqrt{Q}\nabla f(x_t) + \frac{1}{\sqrt{Q}\eta_l|S||U|\bar{\alpha}_t}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] &= Q\mathbb{E}\left[\left\|\nabla f(x_t) + \frac{1}{Q\eta_l|S||U|\bar{\alpha}_t}\sum_{s\in S}\Big(\sum_{u\in U}\Big(-\eta_l\sum_{q\in[Q]}\nabla f_{s,u}(x_{t,q}^{s,u})\cdot\bar{\alpha}_t\Big)\Big)\right\|^2\right] \\ &= Q\mathbb{E}\left[\left\|\frac{1}{Q|S||U|}\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}\nabla f_{s,u}(x_t) - \nabla f_{s,u}(x_{t,q}^{s,u})\right\|^2\right] \\ &\overset{(a1)}{\leq} Q\cdot\frac{1}{Q|S||U|}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}\|\nabla f_{s,u}(x_t) - \nabla f_{s,u}(x_{t,q}^{s,u})\|^2\right] \\ &\overset{(a2)}{\leq} \frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}L^2\mathbb{E}\left[\|x_t - x_{t,q}^{s,u}\|^2\right] \\ &\overset{(a3)}{\leq} L^2 5Q^2\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + L^2 30Q^3\eta_l^2\|\nabla f(x_t)\|^2,\end{aligned} \tag{3.10}$$

where we use $\mathbb{E}\left[\|X_1 + ... + X_n\|^2\right] \leq n\mathbb{E}\left[\|X_1\|^2 + ... + \|X_n\|^2\right]$ at (a1), L-smoothness (Assumption 1) at (a2), and Lemma 3 of [154] at (a3).

Secondly, we evaluate the second-order term in Eq. (3.7) as follows:

$$\begin{aligned}&\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s\right\|^2\right] \\ &\leq 3\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s - \tilde{\Delta}_t^s\right\|^2\right] + 3\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s - \bar{\Delta}_t^s\right\|^2\right] + 3\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\bar{\Delta}_t^s\right\|^2\right].\end{aligned} \tag{3.11}$$

This is because $(a+b+c)^2 \leq 3a^2 + 3b^2 + 3c^2$ holds when $a = A - B$, $b = B - C$, $c = C$ for all vector A, B, C. We can bound the expectation in the last term of

Eq. (3.11) as follows:

$$
\begin{aligned}
\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\bar{\Delta}_t^s\right\|^2\right] &= \mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}(\eta_l\sum_{q\in[Q]}g_{t,q}^{s,u}\cdot\bar{\alpha}_t)\right\|^2\right] \\
&\overset{(a4)}{=} \eta_l^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}\nabla f_{s,u}(x_{t,q}^{s,u})\cdot\bar{\alpha}_t\right\|^2 + \left\|\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}(\nabla f_{s,u}(x_{t,q}^{s,u})-g_{t,q}^{s,u})\cdot\bar{\alpha}_t\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] + \frac{\eta_l^2\bar{\alpha}_t^2}{|S|^2|U|^2}\mathbb{E}\left[\left\|\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}(\nabla f_{s,u}(x_{t,q}^{s,u})-g_{t,q}^{s,u})\right\|^2\right] \\
&\overset{(a5)}{\leq} \mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] + \frac{\eta_l^2\bar{\alpha}_t^2}{|S|^2|U|^2}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}\left\|\nabla f_{s,u}(x_{t,q}^{s,u})-g_{t,q}^{s,u}\right\|^2\right] \\
&\leq \mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] + \frac{\eta_l^2\bar{\alpha}_t^2}{|S|^2|U|^2}\sum_{s\in S}\sum_{u\in U}\sum_{q\in[Q]}\sigma_l^2 \\
&\leq \mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] + \frac{\eta_l^2\bar{\alpha}_t^2}{|S||U|}Q\sigma_l^2, \qquad (3.12)
\end{aligned}
$$

where we use $\mathbb{E}\left[\|X\|^2\right] = \mathbb{E}\left[\|X-\mathbb{E}\left[X\right]\|^2\right]+\|\mathbb{E}\left[X\right]\|^2$ at (a4), and $\mathbb{E}\left[\|X_1+...+X_n\|^2\right] \leq \mathbb{E}\left[\|X_1\|^2+...+\|X_n\|^2\right]$ when $\forall i,j, i\neq j$, X_i and X_j are independent and $\mathbb{E}\left[X_i\right] = 0$ and Assumption 2 at (a5).

Lastly, combining all of above, we have

$$
\begin{aligned}
\mathbb{E}\left[f(x_{x+1})\right] \leq\ & f(x_t) + \eta_g \langle \nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s - \tilde{\Delta}_t^s)\right]\rangle + \eta_g \langle \nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\tilde{\Delta}_t^s - \bar{\Delta}_t^s)\right]\rangle \\
& - \frac{1}{2}\eta_g\eta_l\bar{\alpha}_t Q\|\nabla f(x_t)\|^2 - \frac{1}{2}\frac{\eta_g}{Q|S|^2|U|^2\eta_l\bar{\alpha}_t}\mathbb{E}\left[\left\|\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] \\
& + \frac{\eta_g\eta_l\bar{\alpha}_t}{2}\left(L^2 5Q^2\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + L^2 30Q^3\eta_l^2\|\nabla f(x_t)\|^2\right) \\
& + \frac{3}{2}L\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s - \tilde{\Delta}_t^s\right\|^2\right] + \frac{3}{2}L\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\tilde{\Delta}_t^s - \bar{\Delta}_t^s\right\|^2\right] + \frac{3}{2}L\eta_g^2\frac{1}{|S|^2|U|^2}\mathbb{E}\left[\left\|\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] \\
& + \frac{3}{2}\frac{\eta_g^2\eta_l^2\bar{\alpha}_t^2 LQ\sigma_l^2}{|S||U|} + \frac{1}{2}\frac{L\eta_g^2\sigma^2C^2d}{|S|^2|U|^2} \\
=\ & f(x_t) - \left(\frac{1}{2}\eta_l\eta_g\bar{\alpha}_t Q - \frac{\eta_g\eta_l\bar{\alpha}_t}{2}L^2 30Q^3\eta_l^2\right)\|\nabla f(x_t)\|^2 \\
& - \left(\frac{1}{2}\frac{\eta_g}{Q|S|^2|U|^2\eta_l\bar{\alpha}_t} - \frac{3}{2}L\eta_g^2\frac{1}{|S|^2|U|^2}\right)\mathbb{E}\left[\left\|\sum_{s\in S}\breve{\Delta}_t^s\right\|^2\right] \\
& + \frac{\eta_g\eta_l\bar{\alpha}_t}{2}L^2 5Q^2\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + \frac{3}{2}\frac{\eta_g^2\eta_l^2\bar{\alpha}_t^2 LQ\sigma_l^2}{|S||U|} + \frac{1}{2}\frac{L\eta_g^2\sigma^2C^2d}{|S|^2|U|^2} \\
& + \underbrace{\eta_g \langle \nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s - \tilde{\Delta}_t^s)\right]\rangle + \eta_g \langle \nabla f(x_t), \mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\tilde{\Delta}_t^s - \bar{\Delta}_t^s)\right]\rangle}_{A_1} \\
& + \underbrace{\frac{3}{2}L\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s - \tilde{\Delta}_t^s\right\|^2\right] + \frac{3}{2}L\eta_g^2\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\tilde{\Delta}_t^s - \bar{\Delta}_t^s\right\|^2\right]}_{A_2} \\
\overset{(a6)}{\leq}\ & f(x_t) - \eta_l\eta_g\bar{\alpha}_t Q\left(\frac{1}{2} - 15L^2Q^2\eta_l^2\right)\|\nabla f(x_t)\|^2 \\
& + \frac{\eta_g\eta_l\bar{\alpha}_t}{2}L^2 5Q^2\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + \frac{3}{2}\frac{\eta_g^2\eta_l^2\bar{\alpha}_t^2 LQ\sigma_l^2}{|S||U|} + \frac{1}{2}\frac{L\eta_g^2\sigma^2C^2d}{|S|^2|U|^2} + A_1 + A_2 \\
\overset{(a7)}{\leq}\ & f(x_t) - c\eta_l\eta_g\bar{\alpha}_t Q\|\nabla f(x_t)\|^2 + \frac{\eta_g\eta_l\bar{\alpha}_t}{2}L^2 5Q^2\eta_l^2(\sigma_l^2 + 6Q\sigma_g^2) + \frac{3}{2}\frac{\eta_g^2\eta_l^2\bar{\alpha}_t^2 LQ\sigma_l^2}{|S||U|} + \frac{1}{2}\frac{L\eta_g^2\sigma^2C^2d}{|S|^2|U|^2} + A_1 + A_2
\end{aligned}
\tag{3.13}
$$

where (a6) follows from $\left(\frac{1}{2}\frac{\eta_g}{Q|S|^2|U|^2\eta_l\bar{\alpha}_t} - \frac{3}{2}L\eta_g^2\frac{1}{|S|^2|U|^2}\right) \geq 0$ if $\eta_g\eta_l \leq \frac{1}{3QL\bar{\alpha}_t}$ and replacing the last terms with A_1 and A_2, and (a7) holds because there exists a constant $c > 0$ satisfying $\left(\frac{1}{2} - 15L^2Q^2\eta_l^2\right) > c > 0$ if $\eta_l < \frac{1}{\sqrt{30}QL}$. Divide both sides of (3.13) by $c\eta_l\eta_g Q\bar{\alpha}_t$, sum over t from 0 to $T-1$, divide both sides by T,

and rearrange, we have

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq{}& \frac{1}{cT\eta_g\eta_l Q}\left(\mathbb{E}\left[\frac{f(x_0)}{\bar{\alpha}_0}\right]-\mathbb{E}\left[\frac{f(x_T)}{\bar{\alpha}_{T-1}}\right]\right)\\ &+\frac{5}{2c}L^2Q\eta_l^2(\sigma_l^2+6Q\sigma_g^2)+\frac{3L\eta_g\eta_l\sigma_l^2}{2c|S||U|}\frac{1}{T}\sum_{t=0}^{T-1}\bar{\alpha}_t+\frac{L\eta_g\sigma^2C^2d}{2c\eta_l Q|S|^2|U|^2}\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\bar{\alpha}_t}\\ &+\underbrace{\frac{1}{c\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\bar{\alpha}_t}\mathbb{E}\left[\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s-\tilde{\Delta}_t^s)\right]\rangle+\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\tilde{\Delta}_t^s-\bar{\Delta}_t^s)\right]\rangle\right]}_{B_1}\\ &+\underbrace{\frac{3L\eta_g}{2c\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{\bar{\alpha}_t}\left(\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s-\tilde{\Delta}_t^s\right\|^2\right]+\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\tilde{\Delta}_t^s-\bar{\Delta}_t^s\right\|^2\right]\right)}_{B_2}.\end{aligned} \tag{3.14}$$

Let $\max_{s\in S,u\in U,t\in[T]}\left(\frac{C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]}g_{t,q}^{s,u}\right]\|\right)}\right)$ be $\bar{C}$, and $\min_{s\in S,u\in U,t\in[T]}\left(\frac{C}{\max\left(C,\eta_l\|\mathbb{E}\left[\sum_{q\in[Q]}g_{t,q}^{s,u}\right]\|\right)}\right)$ be $\underline{\mathrm{C}}$, $\frac{1}{|S|}\underline{\mathrm{C}}\leq\bar{\alpha}_t\leq\frac{1}{|S|}\bar{C}$ since $\bar{\alpha}_t$'s definition and $\frac{1}{|S|}\sum_{s\in S}w_{s,u}=\frac{1}{|S|}$. Using this, (3.14) is evaluated as follows:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \leq{}& \frac{1}{cT\eta_g\eta_l Q|S|}\left(\mathbb{E}\left[\frac{f(x_0)}{\underline{\mathrm{C}}}\right]-\mathbb{E}\left[\frac{f(x_T)}{\bar{C}}\right]\right)\\ &+\frac{5}{2c}L^2Q\eta_l^2(\sigma_l^2+6Q\sigma_g^2)+\frac{3\bar{C}L\eta_g\eta_l\sigma_l^2}{2c|S|^2|U|}+\frac{L\eta_g\sigma^2C^2d}{2c\underline{\mathrm{C}}\eta_l Q|S||U|^2}\\ &+\underbrace{\frac{|S|}{c\underline{\mathrm{C}}\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\Delta_t^s-\tilde{\Delta}_t^s)\right]\rangle+\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}(\tilde{\Delta}_t^s-\bar{\Delta}_t^s)\right]\rangle\right]}_{B_1}\\ &+\underbrace{\frac{3L\eta_g|S|}{2c\underline{\mathrm{C}}\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\left(\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\Delta_t^s-\tilde{\Delta}_t^s\right\|^2\right]+\mathbb{E}\left[\left\|\frac{1}{|S||U|}\sum_{s\in S}\tilde{\Delta}_t^s-\bar{\Delta}_t^s\right\|^2\right]\right)}_{B_2}.\end{aligned} \tag{3.15}$$

B_1 and B_2 is upper-bounded as follows:

$$\begin{aligned}B_1 &\leq \frac{|S|}{c\underline{\mathrm{C}}\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\left|\Delta_t^{s,u}-\tilde{\Delta}_t^{s,u}\right|\right]\rangle+\langle\nabla f(x_t),\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\left|\tilde{\Delta}_t^{s,u}-\bar{\Delta}_t^{s,u}\right|\right]\rangle\right]\\ &\leq \frac{G^2}{c\underline{\mathrm{C}}|U|}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}\left(\left|\alpha_t^{s,u}-\tilde{\alpha}_t^{s,u}\right|+\left|\tilde{\alpha}_t^{s,u}-\bar{\alpha}_t\right|\right)\right],\\ B_2 &\overset{(a8)}{\leq} \frac{3L\eta_g|S|}{2c\underline{\mathrm{C}}\eta_l Q}\frac{1}{T}\sum_{t=0}^{T-1}\left(\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\left|\Delta_t^{s,u}-\tilde{\Delta}_t^{s,u}\right|^2\right]+\mathbb{E}\left[\frac{1}{|S||U|}\sum_{s\in S}\sum_{u\in U}\left|\tilde{\Delta}_t^{s,u}-\bar{\Delta}_t^{s,u}\right|^2\right]\right)\\ &\leq \frac{3L\eta_g\eta_l QG^2}{2c\underline{\mathrm{C}}|U|}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\sum_{s\in S}\sum_{u\in U}\left(\left|\alpha_t^{s,u}-\tilde{\alpha}_t^{s,u}\right|^2+\left|\tilde{\alpha}_t^{s,u}-\bar{\alpha}_t\right|^2\right)\right]\end{aligned} \tag{3.16}$$

with

$$\left|\Delta_t^{s,u} - \tilde{\Delta}_t^{s,u}\right| \leq \eta_l Q G \left|\alpha_t^{s,u} - \tilde{\alpha}_t^{s,u}\right|,$$
$$\left|\tilde{\Delta}_t^{s,u} - \bar{\Delta}_t^{s,u}\right| \leq \eta_l Q G \left|\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t\right|, \tag{3.17}$$

by Assumption 3 of globally bounded gradients, and we use the same technique at (a8) as (a1).

□

Remark 2. The first three terms recover the standard convergence bound up to constants for FedAVG [89] when considering participants in FL as user-silo pairs (i.e., we have $|S||U|$ participants). The asymptotic bound is $O(\frac{1}{\sqrt{|S||U|QT}} + \frac{1}{T})$. Theorem 6 achieves this when we choose the global and local learning rates $\eta_g = |S|\sqrt{|S||U|Q}$ and $\eta_l = \frac{1}{\sqrt{TQL}}$, respectively. This requires a learning rate $|S|$ times larger than the usual FedAVG with $|S||U|$ participants, which can be interpreted as coming from the constraint on the weights W. The fourth term is the convergence overhead due to Gaussian noise addition, and the last fifth and sixth terms are the overhead due to bias from the clippings. If both the noise and the clipping bias are zero, the convergence rate is asymptotically the same as the FedAVG convergence rate.

Remark 3. The fourth term, accounting for the overhead due to noise, differs slightly from DP-FedAVG. This term is inversely proportional to $|S||U|^2$. As highlighted in the previous remark, if the global learning rate is set as $\eta_g = |S|\sqrt{|U|Q}$ in ULDP-AVG, this term becomes proportional to $\sqrt{|U|}/|U|^2$, consistent with the case of DP-FedAVG with $|U|$ participants.

Remark 4. The fifth and sixth terms correspond to the overhead due to the clipping biases $|\alpha_t^{s,u} - \tilde{\alpha}_t^{s,u}|$ and $|\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t|$, respectively. The quantity $|\alpha_t^{s,u} - \tilde{\alpha}_t^{s,u}|$ represents the local gradient variance across all users in all silos and can be made zero by full-batch gradient descent. The noteworthy term is $|\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t|$. As discussed in the analysis of [153], this term is influenced by the structure of the neural network and data heterogeneity. It roughly quantifies how much the magnitudes of all gradients deviate from the global mean gradient. We may be able to minimize these values by selecting appropriate weights $\mathbf{W}$, guided by the

following optimization problem:

$$\min_{\mathbf{W}} \sum_{s\in S}\sum_{u\in U} |\tilde{\alpha}_t^{s,u} - \bar{\alpha}_t|, \text{ s.t., } w_{s,u} > 0, \forall u, \sum_{s\in S} w_{s,u} = 1$$
$$\left(= \sum_{s\in S}\sum_{u\in U} \left| w_{s,u}C_{s,u} - \frac{1}{|S||U|}\sum_{s'\in S}\sum_{u'\in U} w_{s',u'}C_{s',u'} \right| \right)$$

where $C_{s,u} := \frac{C}{\max\left(C, \eta_l \| \mathbb{E}\left[\sum_{q\in[Q]} g_{t,q}^{s,u}\right] \|\right)}$.
However, determining the optimal weights is challenging because we cannot predict the gradients' norms in advance, which could also cause another privacy issue.

3.4 Private weighting protocol

In considering the bias described in Remark 4, we employed uniform clipping weights in the ULDP-AVG algorithm, i.e., for any $s \in S$ and $u \in U$, we set $w_{s,u} = 1/|S|$, as a feasible solution to the problem without privacy violation. However, for a more sophisticated solution, we propose the following weighting strategy. We set a weight $w_{s,u}^{opt}$ for $C_{s,u}$ according to the number of records, following the heuristic that a gradient computed from a large number of records yields a better estimation that is closely aligned with the average. This results in smaller bias. That is, let $n_{s,u}$ be the number of records for user u in silo s, we set the weight as follows:

$$w_{s,u}^{opt} := \frac{n_{s,u}}{\sum_{s\in|S|} n_{s,u}}. \tag{3.18}$$

We empirically demonstrate the effectiveness of this strategy in our experiments. However, the crucial question arises: *how can this be implemented without violating privacy?*

For the aforementioned better weighting strategy, a central server could aggregate histograms encompassing the user population (number of records per user) within each silo's dataset. Subsequently, the server could compute the appropriate weights for each silo and distribute these weights back to the respective silos. However, this approach raises significant privacy concerns, leading to a privacy breach as the silo histograms are directly shared with the server. Moreover, when the server broadcasts the weights back to the silos, it enables an estimation of the entire histogram of users across all the silos, posing a similar privacy risk against

other silos. In essence, privacy protection necessitates preserving confidentiality in both of these directions. This is challenging because additive homomorphic encryption techniques, such as the Paillier cryptosystem, cannot handle inverses to compute weights as in Eq. (3.18), and the raw weights are disclosed to the party with the secret key when encrypting the weights.

To address this privacy issue, we designed a novel private weighting protocol to securely aggregate the user histograms from silos, compute the per-user clipping weight for each user in each silo, and aggregate the weighted sum from all silos. The protocol leverages well-established cryptographic techniques, including secure aggregation [66, 149], the Paillier cryptosystem [155], and multiplicative blinding [156]. Intuitively, the protocol employs multiplicative blinding to hide user histograms against the server while allowing the server to compute inverses of blinded histograms to compute the weights (Eq. (3.18)). Subsequently, the server employs the Paillier encryption to conceal the inverses of blinded histograms because the silo knows the blinded masks. This also enables the server and silos to compute private weighted sum aggregation with its additive homomorphic property.

The details of the private weighting protocol are explained in Protocol 1. The protocol consists of a setup phase, which is executed only once during the entire training process, and a weighting phase, which is executed in each round of training. In the setup phase, as depicted in (a-c) of Protocol 1, the server generates a key-pair for Paillier encryption, while the silos establish shared random seeds through a Diffie–Hellman (DH) key exchange via the server. Subsequently, in steps of (d-f), the blinded inverses of the user histogram are computed. In the weighting phase, (a) the server prepares the encrypted weights, (b) the silos compute user-level weighted model deltas in the encrypted world, and (c) the server recovers the aggregated value. It is important to note that in the Paillier cryptosystem, the plaintext data x exists within the additive group modulo n, while encrypted data (denoted as $Enc_{\mathrm{p}}(x)$) belongs to the multiplicative group modulo n^2 with an order of n. The system allows for operations such as addition of ciphertexts and scalar multiplication and addition on ciphertexts.

Note that our assumption is that the results of user-level sub-sampling are allowed to be open to the aggregation server. This is also the case in Protocol 1, but it could be hidden by combining the two-party verifiable sampling scheme with 1-out-of-P Oblivious Transfer as described in Chapter 5, which would cause

Protocol 1 Private Weighting Protocol

Inputs: Silo $s \in S$ that holds an dataset with $n_{s,u}$ records for each user $u \in U$. $\mathcal{A}$ is central aggregation server. $N_{\max}$ is upper bound on the number of records per user, e.g., 2000. P is precision parameter, e.g, 10^{-10}. λ is security parameter, e.g., 3072-bit security.

1. **Setup.**
   (a) $\mathcal{A}$ generates Paillier keypairs (PK, SK) with the given security parameter λ and sends the public key PK to all silos. All silos s generate DH keypairs (pk_s, sk_s) with the same parameter λ and transmit their respective public key pk_s to $\mathcal{A}$. Both $\mathcal{A}$ and all silos compute C_{LCM}, which is the least common multiple of all integers up to $N_{\max}$. The modulus n included in PK is used for the finite field $\mathbb{F}_n$ by $\mathcal{A}$ and all silos.
   (b) After receiving all pk_s, $\mathcal{A}$ broadcasts all DH public keys pk_s to all s. All s compute shared keys $sk_{s,s'}$ from sk_s and received public keys $pk_{s'}$ for all $s' \in S$.
   (c) Silo 0 ($\in S$) generates a random seed R and encrypts R using $sk_{0,s'}$ to obtain $Enc(R)$ and sends $Enc(R)$ to s' via $\mathcal{A}$ for all s'. All $s \in S \setminus 0$ receive and decrypt $Enc(R)$ with $sk_{s,0}$ and get R as a shared random seed.
   (d) All s generate multiplicative blind masks $r_u \in \mathbb{F}_n$ with the same R and compute blinded histograms as $B(n_{s,u}) \equiv r_u n_{s,u} \pmod n$ for all $u \in U$.
   (e) All s generate pair-wise additive masks $r^u_{s,s'} \in \mathbb{F}_n$ employing $sk_{s,s'}$ for all s' and u, with $r^u_{s,s'} = r^u_{s',s}$. Subsequently, they calculate the doubly blinded histogram as $B'(n_{s,u}) \equiv B(n_{s,u}) + \sum_{s<s'} r^u_{s,s'} - \sum_{s>s'} r^u_{s,s'} \pmod n$. All s send $B'(n_{s,u})$ to $\mathcal{A}$. $\mathcal{A}$ aggregates these contributions to compute $B(N_u) \equiv \sum_{s\in S} B'(n_{s,u}) \pmod n$ for each u, denoting $N_u = \sum_{s\in S} n_{s,u}$.
   (f) $\mathcal{A}$ computes the inverse of $B(N_u)$ as $B_{\text{inv}}(N_u) = B(N_u)^{-1}$ for each u. This is the multiplicative inverse on $\mathbb{F}_n$, which is efficiently computed by the Extended Euclidean algorithm.
2. **Weighting for each training round t.**
   (a) $\mathcal{A}$ encrypts $B_{\text{inv}}(N_u)$ using Paillier's public key PK, resulting in $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ for all u. If user-level sub-sampling is required, the server performs Poisson sampling with a given probability q for each user before the encryption. For non-selected users, $B_{\text{inv}}(N_u)$ is set to 0. If we require user-level sub-sampling, we perform Poisson sampling with given probability q on the server for each user before the Paillier's encryption and set $B_{\text{inv}}(N_u) = 0$ for all users not selected. Subsequently, $\mathcal{A}$ broadcasts all $Enc_{\text{p}}(B_{\text{inv}}(N_u))$ to all silos.
   (b) In each s, following the approach of ULDP-AVG, the clipped model delta $\tilde{\Delta}^{s,u}_t$ is computed for each user u. The weighted clipped model delta is then calculated as $Enc_{\text{p}}(\tilde{\Delta}^{s,u}_t) = \textsc{Encode}(\tilde{\Delta}^{s,u}_t, P, n) n_{s,u} r_u C_{\text{LCM}} Enc_{\text{p}}(B_{\text{inv}}(N_u))$.
   Let the Gaussian noise be z^s_t, we then compute $z'_s = \textsc{Encode}(z^s_t, P, n) C_{\text{LCM}}$. Note that we need to approximate real number $\tilde{\Delta}^{s,u}_t$ and z^s_t on a finite field using Encode (described in Algorithm 8). Lastly, we compute the summation $Enc_{\text{p}}(\Delta^s_t) = \sum_{u\in U} Enc_{\text{p}}(\tilde{\Delta}^{s,u}_t) + z'_s$.
   (c) In each s, random pair-wise additive masks are generated, and secure aggregation is performed on $Enc_{\text{p}}(\Delta^s_t)$ mirroring the steps in 1.(f). Then, $\mathcal{A}$ gets $\sum_{s\in S} Enc_{\text{p}}(\Delta^s_t)$. $\mathcal{A}$ decrypts it with Paillier's secret key SK and decodes it by $\textsc{Decode}(\sum_{s\in S} \Delta^s_t, P, C_{\text{LCM}}, n)$ and recovers the aggregated value.
   (d) Steps 2.(a) through 2.(c) are repeated for each training round.

Algorithm 8 ENCODE and DECODE functions in the private weighting protocol

procedure ENCODE(x, P, n) ▷ e.g., $P = 10^{-10}$
  /* to turn floating point into fixed point */
  $x \leftarrow x/P$ ▷ compute as floating point
  $x \leftarrow x$ as integer
  /* to map integer $\mathbb{Z}$ into finite field $\mathbb{F}_n$*/
  $x \leftarrow x \,(\mathrm{mod}\, n)$
  return x
end procedure

procedure DECODE($x, P, C_{\mathrm{LCM}}, n$)
  /* to map finite field $\mathbb{F}_n$ number into integer $\mathbb{Z}$ */
  if $x > n//2$ **then** ▷ // means integer division
    $x \leftarrow x - n$
  else
    $x \leftarrow x$
  end if
  /* to remove C_{LCM} factor*/
  $x \leftarrow x/C_{\mathrm{LCM}}$ ▷ compute as floating point
  /* to recover original magnitude */
  $x \leftarrow xP$
  return x
end procedure

extra computation costs. As an overview, for each user u, the server creates $P-1$ dummy data $Enc_{\mathrm{p}}(0)$ for $Enc_{\mathrm{p}}(B_{\mathrm{inv}}(N_u))$ described in 2.(a) of Protocol 1. When the client performs OT on this data, the selection probability of $Enc_{\mathrm{p}}(B_{\mathrm{inv}}(N_u))$ is $\frac{1}{P}$ and that of $Enc_{\mathrm{p}}(0)$ is $\frac{P-1}{P}$. The selection of $Enc_{\mathrm{p}}(B_{\mathrm{inv}}(N_u))$ means that the user is not sampled by the user-level sub-sampling. In this way, the server does not know which data was retrieved by the client from the OT, and the client cannot know the sampling result due to the Paillier encryption. However, the expressed probability is likely to be less strict because it can only represent discrete probability distributions just like the proposed method in Chapter 5.

Next, we provide a theoretical analysis of this private weighting protocol (Protocol 1) in terms of correctness and privacy.

3.4.1 Correctness

The protocol must compute the correct result as the same result as the non-secure method. To this end, we consider the correctness of the aggregated data obtained in each round. Let $\sum_{s\in S}\Delta_t^s$ with the non-secure method be Δ and the one with the Protocol 1 be Δsec, our goal is formally stated as $\Pr[|\Delta-\Delta_{\text{sec}}|_\infty > P] < negl$, where P is a precision parameter and $negl$ signifies a negligible value.

Initially, with regards to secure aggregation, the additive pair-wise masks cancel out as shown in [66]. For the difference, silos must participate in any rounds in the cross-silo FL. When collecting the histogram, additive masks are canceled out as follows:

$$B(N_u) = \sum_{s\in S} B(n_{s,u}) + \sum_{s<s'} \underbrace{(r^u_{s,s'} - r^u_{s',s})}_{\text{canceled out}} - \sum_{s>s'} \underbrace{(r^u_{s,s'} - r^u_{s',s})}_{\text{canceled out}}$$

The same applies to secure aggregation for model delta. Note that the mask is not directly added to the ciphertext (within the multiplication group of n^2); instead, scalar addition within $\mathbb{F}_n$ that takes advantage of the homomorphic property of the Paillier cryptosystem is utilized. As secure aggregation doesn't result in any degradation of the aggregation outcome when all terms are within $\mathbb{F}_n$, our focus shifts to other components. Note that there are errors due to the handling of fixed-point numbers on a finite field.

From the protocol description, $Enc_{\text{p}}(\Delta_{\text{sec}})$ is analyzed as follows:

$$\begin{aligned}
&Enc_{\text{p}}(\Delta_{\text{sec}}) = \sum_{s\in S} Enc_{\text{p}}(\Delta_t^s) = \sum_{s\in S}\sum_{u\in U} Enc_{\text{p}}(\tilde{\Delta}_t^{s,u}) + z'_s \\
&= \sum_{s\in S}\sum_{u\in U} E_t^{s,u} n_{s,u} r_u C_{\text{LCM}} Enc_{\text{p}}(B_{\text{inv}}(N_u)) + Z_t^s C_{\text{LCM}} \\
&\overset{(1)}{=} Enc_{\text{p}}(\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} r_u C_{\text{LCM}} B_{\text{inv}}(N_u))) + Z_t^s C_{\text{LCM}}) \\
&= Enc_{\text{p}}(\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} r_u C_{\text{LCM}} (r_u N_u)^{-1}) + Z_t^s C_{\text{LCM}})) \\
&\overset{(2)}{=} Enc_{\text{p}}(\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} C_{-N_u}) + Z_t^s C_{\text{LCM}}))
\end{aligned}$$

where C_{-N_u} is the result of modular multiplication between C_{LCM} and the modular multiplicative inverse of N_u, $E_t^{s,u} = \textsc{Encode}(\tilde{\Delta}_t^{s,u}, P, n)$ and $Z_t^s = \textsc{Encode}(z_t^s, P, n)$. Equation (1) is because all of B_{inv}, $E_t^{s,u}$, $n_{s,u}$, r_u and $C_{\text{LCM}} \in \mathbb{F}_n$. At (2), the

modulo inverse of r_u is canceled out because $r_u \in \mathbb{F}_n$ and r_u almost always has a modulo inverse. However, this does not hold when r_u and n are not coprime. Let p and q be two large primes used in the Paillier cryptosystem, such that $n = pq$, then the probability of a random $r_u \in \mathbb{F}_n$ and n are not coprime is

$$\frac{n-1-\phi(n)}{n-1} = \frac{n-1-(p-1)(q-1)}{n-1}, \tag{3.19}$$

where ϕ is Euler's totient function. This probability is negligibly small when n is sufficiently large. In the case of user-level sub-sampling, $B_{\text{inv}}(N_u)$ is set 0 and we see that only the model delta for user u is all zeros, which produces exactly the same result as if the unselected users did not participate in the training round. The important condition is that if $N_u \leq N_{\max}$ and N_u is a factor of C_{LCM}, C_{LCM}/N_u is always divisible on $\mathbb{Z}$ and the result is the same as on $\mathbb{F}_n$. Also if $\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} C_{-N_u}) + Z_t^s C_{\text{LCM}}) \in \mathbb{Z}$ is smaller than n, it yields the same results on $\mathbb{Z}$ and $\mathbb{F}_n$. Hence, when these conditions are satisfied, decrypted value $\Delta_{\text{sec}} \in \mathbb{F}_n$ obtains the same result on $\Delta_{\text{sec}} \in \mathbb{Z}$. After decryption, we consider $\Delta_{\text{sec}} \in \mathbb{R}$. The final result is decoded by $\textsc{Decode}(\Delta_{\text{sec}}, P, C_{\text{LCM}}, n) \in \mathbb{R}$ as follows:

$$\textsc{Decode}(\Delta_{\text{sec}}, P, C_{\text{LCM}}, n) = \sum_{s\in S}(\sum_{u\in U}(\frac{n_{s,u}}{N_u}\tilde{\Delta}'_{s,u,t}) + Z'_{s,t})$$

where, from Algorithm 8, $\tilde{\Delta}'_{s,u,t}$ is the same as the result of computing $\tilde{\Delta}_t^{s,u}$ in fixed-point with precision P, and $Z'_{s,t}$ is the same as z_t^s.

Therefore, the correctness of these calculations is satisfied when two conditions (1) $N_u \leq N_{\max}$ and (2) $\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} C_{-N_u}) + Z_t^s C_{\text{LCM}}) < n$, are satisfied. To satisfy (1), $N_{\max}$ must be sufficiently large, and a larger $N_{\max}$ leads to a larger C_{LCM}. Hence, when we take n is large, these conditions can be satisfied unless the parameters or noise take on unrealistic values.

For example, suppose the range of the noise and aggregated model parameters is $[-10^{10}, 10^{10}]$, $P = 10^{-10}$, $N_{\max} = 2000$ and λ is 3072-bit security, we have $n > 10^{924}$ by Paillier cryptosystem, $E_t^{s,u} < 10^{20}$, $Z_t^s < 10^{20}$ and $C_{\text{LCM}} < 10^{867}$. Then, $\sum_{s\in S}(\sum_{u\in U}(E_t^{s,u} n_{s,u} C_{-N_u}) + Z_t^s C_{\text{LCM}}) < 10^{888} < n$ and we satisfy the condition (2). C_{LCM} grows exponentially with respect to $N_{\max}$. One possible solution for this is that we can make it very small by limiting the number of records per user to several values. For example, $\{10^1, 10^2, 10^3, 10^4\}$ then $C_{\text{LCM}} = 10^4$.

3.4.2 Privacy

Privacy in this protocol means that both the central server and the silos do not get more information than is available in the original ULDP-AVG while we perform the better weighting strategy. Our approach relies on a private weighted sum aggregation technique employing the Paillier cryptosystem [157, 107] and secure aggregation [66]. Formal security arguments for these methodologies are available in their respective sources. The protocol is fully compatible with these works because all data exchanged is handled on $\mathbb{F}_n$ including random masks for secure aggregation.

A different view of the server from these basic methods is $B(N_u)$ for all u, which is multiplicatively blinded aggregated histogram. Since $B(n_{s,u})$ for s and u is securely hidden by secure aggregation, only $B(N_u)$ is the meaningful server view. $B(N_u)$ is $r_u N_u \,(\mathrm{mod}\, n)$ and this is randomly distributed on $\mathbb{F}_n$ if r_u is uniformly distributed on $\mathbb{F}_n$. Also, the inverse of $B_{\mathrm{inv}}(N_u)$ is uniformly distributed as well. This is because the multiplication operation in a finite field is closed and bijective. Therefore, it is information-theoretically indistinguishable and private. Such multiplicative blinding has been also used in [156]. The view of the silos is the same as that in [107], even though the contents of the weights are sensitive, and privacy is protected by Paillier encryption. Note that the security of the initial DH key exchange and the security of the Paillier cryptosystem follow the security parameter λ, which is an input to the protocol.

3.5 Experiments

3.5.1 Settings

In this section, we evaluate the privacy-utility trade-offs of the proposed methods (ULDP-AVG/SGD), along with the previously mentioned baselines (ULDP-NAIVE/GROUP-k) and a non-private baseline (FedAVG with two-sided learning rates [89], denoted by DEFAULT). In ULDP-AVG/SGD, we set the weights as $w_{s,u} = 1/|S|$ for all s and u, the one using $w_{s,u}^{opt}$ is referred to as ULDP-AVG-w. Regarding ULDP-GROUP-k, flags $\mathbf{B}$ are generated for existing records to minimize waste on filtered out records, despite the potential privacy concerns. Various values, including the maximum number of user records, the median, 2, and 8, are

tested as group size k and we report GDP using group-privacy conversion of RDP. In cases where k is not a power of 2, the computed ϵ is reported for the largest power of 2 below k, showcasing the lower bound of GDP to underscore that ϵ is large. The hyperparameters, including global and local learning rates η_g, η_l, clipping bound C, and local epoch Q, are set individually for each method. Execution times are measured on macOS Monterey v12.1, Apple M1 Max Chip with 64GB memory with Python 3.9 and 3072-bit security. Most of the results are averaged over 5 runs and the colored area in the graph represents the standard deviation. All of our implementations and experimental settings are available [b].

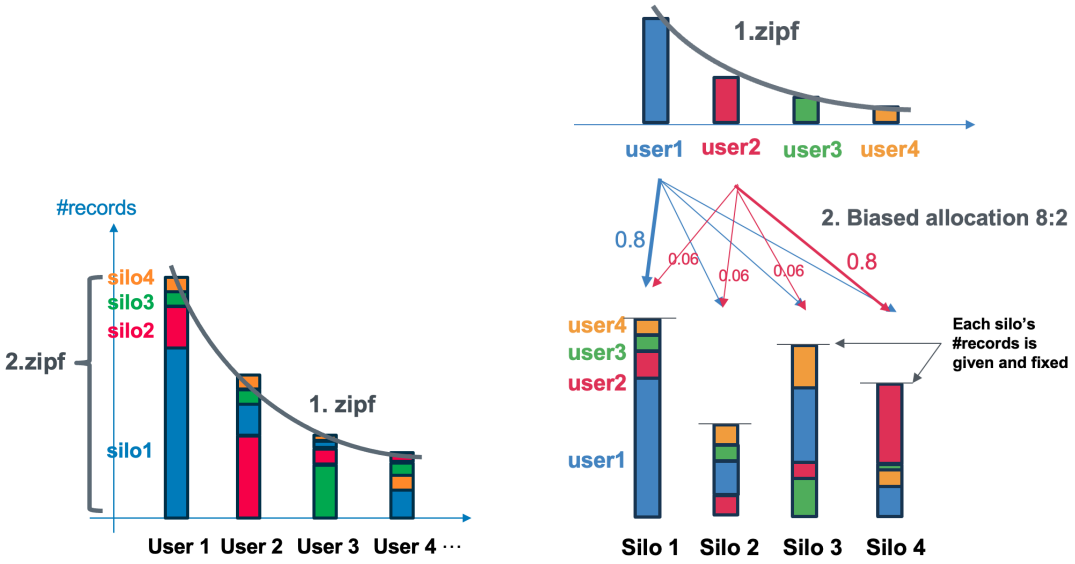
Datasets

Datasets used in the evaluation comprise real-world open datasets, including Credicard [158], well-known image dataset MNIST and two medical datasets, HeartDisease and TcgaBrca from [106], benchmark datasets for cross-silo FL. Creditcard is a popular tabular dataset for credit card fraud detection from Kaggle. We undersample the dataset and use about 25K training data and a neural network with about 4K parameters. For MNIST, we use a CNN model with about 20K parameters, 60K training data and 10K evaluation data, and assign silos and users to all of the training data. For HeartDisease and TcgaBrca, we use the same settings such as number of silos (i.e., 4 and 6), data assignments to the silos, models, loss functions, etc. as shown in the original paper. These two datasets are quite small and the model has less than 100 parameters.

For all datasets, we need to link all records to each user and silo, and how to allocate the records to users and silos is explained below.

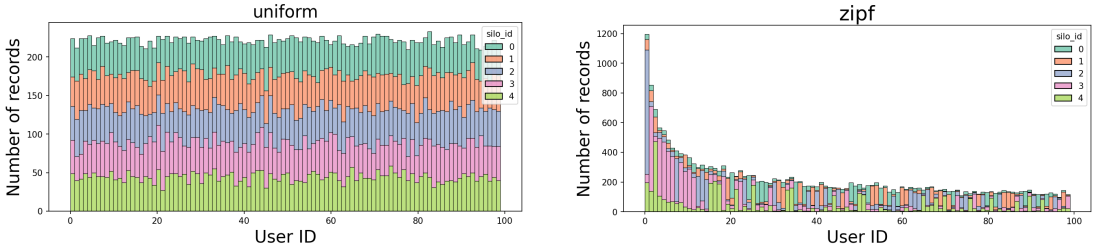
Record allocation for MNIST and Creditcard. We designed two different record distribution patterns, *uniform* and *zipf*, to model how user records are scattered across silos in the MNIST and Creditcard datasets. Both distributions take the number of users $|U|$ and the number of silos $|S|$. It associates each record with a user and a silo. (1) In uniform, every record is assigned to a user with equal probability, and likewise, each record is assigned to a silo with equal probability. (2) Zipf combines two types of Zipf distributions as shown in Figure 3.4a. First, the distribution of the number of records per user follows a Zipf distribution. Then, for each user, the numbers of records are assigned to different silos based

[b] `https://github.com/FumiyukiKato/uldp-fl`



(a) MNIST and Creditcard. (b) HeartDisease and TcgaBrca.

Figure 3.4: Record distribution across users and silos.



(a) uniform. (b) zipf.

Figure 3.5: Example of record allocation on Creditcard.

on another Zipf distribution. Each of the two Zipf distributions takes a parameter α that determines the concentration of the numbers. In the experiments, we used $\alpha = 0.5$ for the first distribution and $\alpha = 2.0$ for the second distribution. This choice is rooted in the observation that the concentration of user records is not as high as the concentration in the silos selected by each user.

For Creditcard and MNIST, the number of silos $|S|$ is fixed at 5. We used 100, 1000 for Creditcard as $|U|$ and 100, 10000 for MNIST. For MNIST, we can require each user to have only 2 labels at most in the non-i.i.d. setting. For example, when $|U| = 100$ and $|S| = 5$, the record distribution of Creditcard dataset is shown in Figure 3.5. The number of records is plotted for each user

and color-coded for each silo.

Record allocation for HeartDisease and TcgaBrca. For the HeartDisease and TcgaBrca datasets, we adopted the same two distributions *uniform* and *zipf* as the above-mentioned ones. In the benchmark datasets HeartDisease and TcgaBrca, all records are already allocated to silos and the number of records of each silo is fixed. Therefore, the design of the user-record allocation is slightly different. (1) In uniform, all records belong to one of the users with equal probability without allocation to silos. (2) In zipf, as shown in Figure 3.4b, the number of records for a user is first generated according to a Zipf distribution, and 80% of the records are assigned to one silo, and the rest to the other silos with equal probability. The priority of the silo is chosen randomly for each user. We used $\alpha = 0.5$ for the parameter of the Zipf. In TcgaBrca, Cox-Loss is used for loss function [106], which needs more than two records for calculating valid loss and we set more than two records for each silo and user for per-user clipping of ULDP-AVG.

Hyperparameters

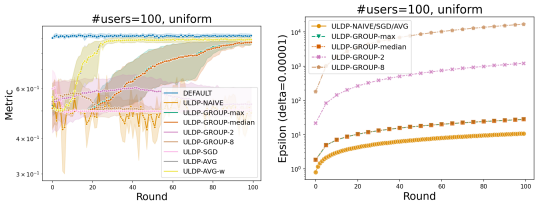
All scripts used in the experiments, including hyperparameters, can be accessed [c].

3.5.2 Results

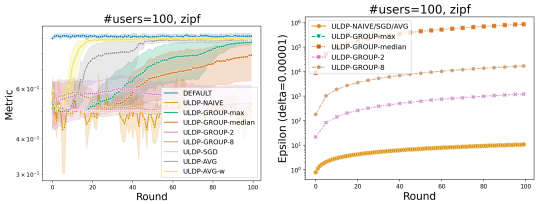
Privacy-utility trade-offs under ULDP. Figures 3.6 show the utility and privacy evaluation results on Creditcard. The average number of records per user (denoted as n) in entire silos and the distribution changes for each figure. All experiments used a fixed noise parameter $\sigma = 5.0$ and $\delta = 10^{-5}$, utility metrics (Accuracy for Creditcard) are displayed on the left side and accumulated privacy consumption ϵ for ULDP are depicted on the right side. Note that the privacy bounds for ULDP-GROUP-k are derived from the local DP-SGD and depend on not only the group size k but also the size of the local training dataset.

Overall, the proposed method ULDP-AVG/SGD achieves competitive utility with fast convergence and high accuracy, while achieving considerably small privacy bounds, which means significantly better privacy-utility trade-offs compared to other baselines. We observe that the baseline method, ULDP-NAIVE, has low

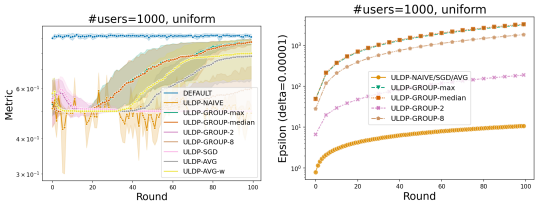
[c] `https://github.com/FumiyukiKato/uldp-fl/blob/main/exp/script/privacy_utility.sh`



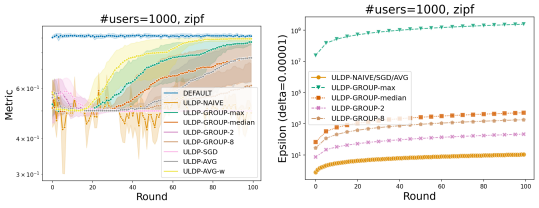
(a) $n \approx 246$ ($|U| = 100$), uniform, Test Accuracy and Privacy.



(b) $n \approx 246$ ($|U| = 100$), zipf, Test Accuracy and Privacy.

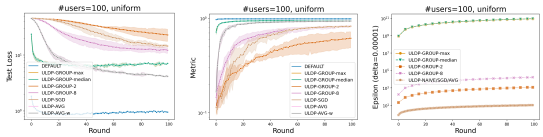


(c) $n \approx 25$ ($|U| = 1000$), uniform, Test Accuracy and Privacy.

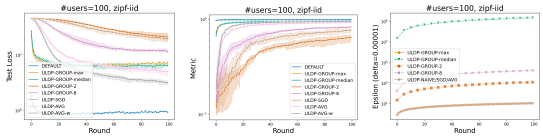


(d) $n \approx 25$ ($|U| = 1000$), zipf, Test Accuracy and Privacy.

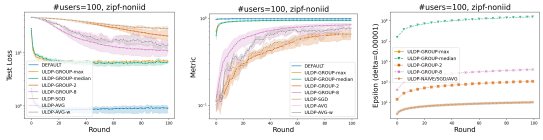
Figure 3.6: Privacy-utility trade-offs on Creditcard dataset: Accuracy (Left), Privacy (Right).



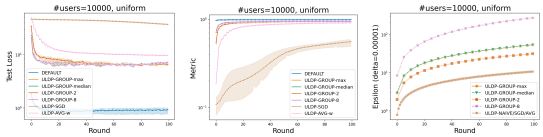
(a) $n \approx 600$ ($|U| = 100$), uniform.



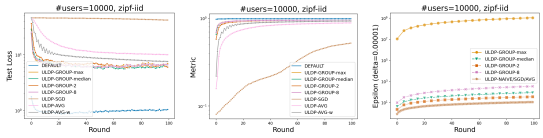
(b) $n \approx 600$ ($|U| = 100$), zipf, iid.



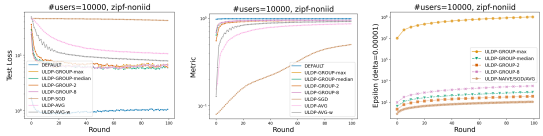
(c) $n \approx 600$ ($|U| = 100$), zipf, non-iid.



(d) $n \approx 6$ ($|U| = 10000$), uniform, iid.



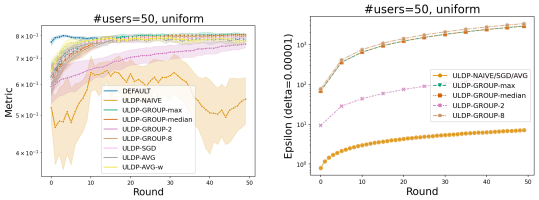
(e) $n \approx 6$ ($|U| = 10000$), zipf, iid.



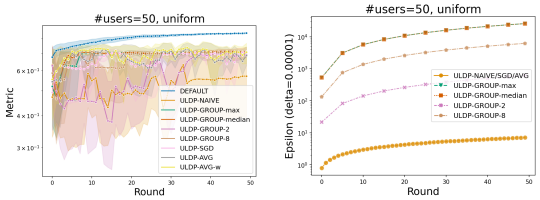
(f) $n \approx 6$ ($|U| = 10000$), zipf, non-iid.

Figure 3.7: Privacy-utility trade-offs on MNIST dataset: Test Loss (Left), Accuracy (Middle), Privacy (Right).

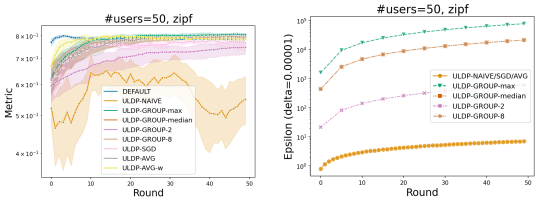
accuracy and that ULDP-GROUP-k requires much larger privacy budgets, which is consistent with the analysis on the conversion of group privacy described earlier. The convergence speed of ULDP-AVG is faster than that of ULDP-SGD, which is the same as that of DP-FedAVG/SGD. Nevertheless, there is still a gap between ULDP-AVG and the non-private method (DEFAULT) in terms of convergence speed and ultimately achievable accuracy, as a price for privacy. Also, as



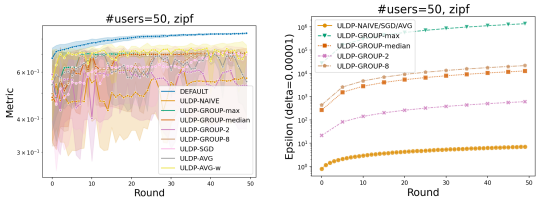
(a) $n \approx 10$ ($|U| = 50$), uniform.



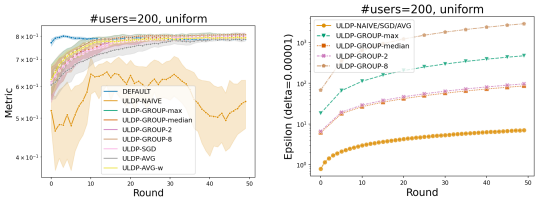
(a) $n \approx 17$ ($|U| = 50$) uniform.



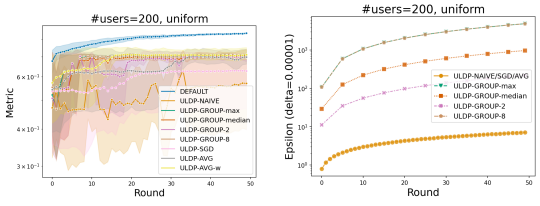
(b) $n \approx 10$ ($|U| = 50$), zipf.



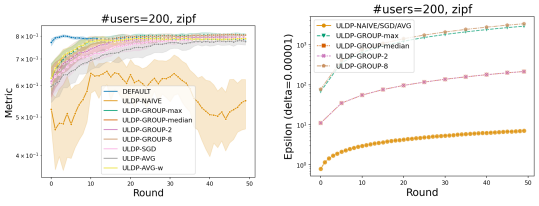
(b) $n \approx 17$ ($|U| = 50$), zipf.



(c) $n \approx 2.5$ ($|U| = 200$), uniform.

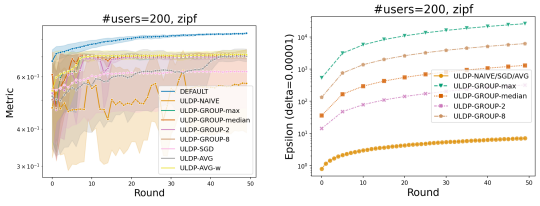


(c) $n \approx 4$ ($|U| = 200$), uniform.



(d) $n \approx 2.5$ ($|U| = 200$), zipf.

Figure 3.8: HeartDisease.



(d) $n \approx 4$ ($|U| = 200$), zipf.

Figure 3.9: TcgaBrca.

shown in Figure 3.6c, for small n (i.e., a large number of users), ULDP-GROUP-max/median show higher accuracy than ULDP-AVG. This is likely due to the overhead from finer datasets at user-level, which increases the bias compared to DP-FedAVG, as seen in the theoretical convergence analysis for ULDP-AVG.

Figure 3.8, 3.9, and 3.7 show privacy-utility trade-offs on HeartDisease, TcgaBrca, and MNIST, respectively. All experiments used a fixed noise parameter (noise multiplier) $\sigma = 5.0$ and $\delta = 10^{-5}$, utility metrics (Accuracy for HeartDisease and MNIST, C-index for TcgaBrca) are plotted on the left side, and accumulated privacy consumption ϵ for ULDP are plotted on the right side. For clarity, the test loss is shown on the left-hand side for MNIST. The average number of records per user (denoted as n) in entire silos and the distribution (uniform/zipf)

changes for each figure.

In all datasets, consistently, ULDP-AVG is competitive in terms of utility, ULDP-AVG-w shows much faster convergence, and ULDP-SGD shows slower convergence. ULDP-NAIVE achieves a low privacy bound; however, its utility is much lower than other methods. The group-privacy methods such as ULDP-GROUP-k show reasonably high utility, especially in settings where n is small. This is because the records to be removed due to the number of records per user being over group size k is small. However, the ULDP privacy bound ULDP-GROUP-k achieves ends up being very large. Note that the privacy bounds for ULDP-GROUP-k are derived from the local DP-SGD and depend on not only the group size k but also the size of the local training dataset. The exceptions are cases where the local data set size is large and the number of records per user is very small as in Figures 3.7d, 3.7e, and 3.7f. In these cases, ULDP-GROUP-2 achieves a reasonably small privacy bound. In other words, if the number of user records is fixed at one or two in the scenario, and the number of training records is large (it is advantageous for ULDP-GROUP because the record-level sub-sampling rate in DP-SGD becomes small), it could be better to use ULDP-GROUP.

The results for the MNIST, non-i.i.d, and $|U| = 100$ case highlight a weak point of ULDP-AVG. Note that non-i.i.d here is at user-level and DEFAULT and ULDP-GROUP are less affected by non-i.i.d because they train per silo rather than per user. As Figure 3.7c shows, the convergence of ULDP-AVG is worse compared to other results. It suggests that ULDP-AVG may emphasize the bad effects of user-level non-i.i.d. distribution that were not an issue with normal cross-silo FL because the gradient is not computed at the user level as in ULDP-AVG. On the other hand, this is less problematic when the number of users is large as shown in Figure 3.7f. This is due to the relatively smaller effect of individual user overfitting caused by non-i.i.d. distribution as the sample size (the number of users) increases.

Effectiveness of the better weighting strategy. To highlight the effectiveness of the better weighting strategy, Figure 3.10 shows the test losses of the Creditcard dataset on different record distributions with ULDP-AVG and ULDP-AVG-w. We present the results with various numbers of silos: 5, 20, and 50. The need for the better weighting strategy is emphasized by the distribution of the records and the number of silos $|S|$. When there are large skews in the

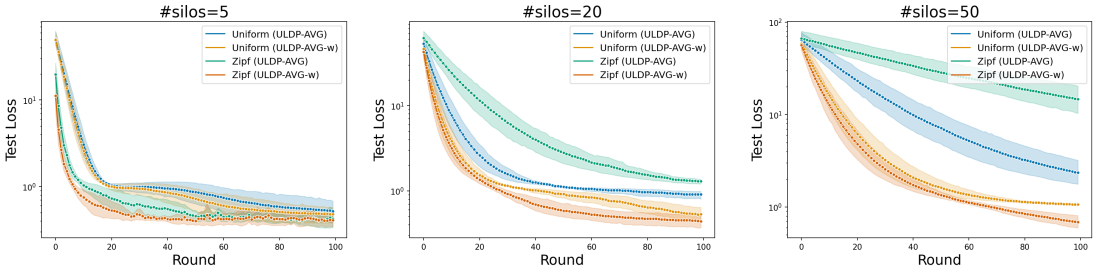


Figure 3.10: Test loss of Creditcard: Weighting method is effective, especially in skewed distribution in many silos.

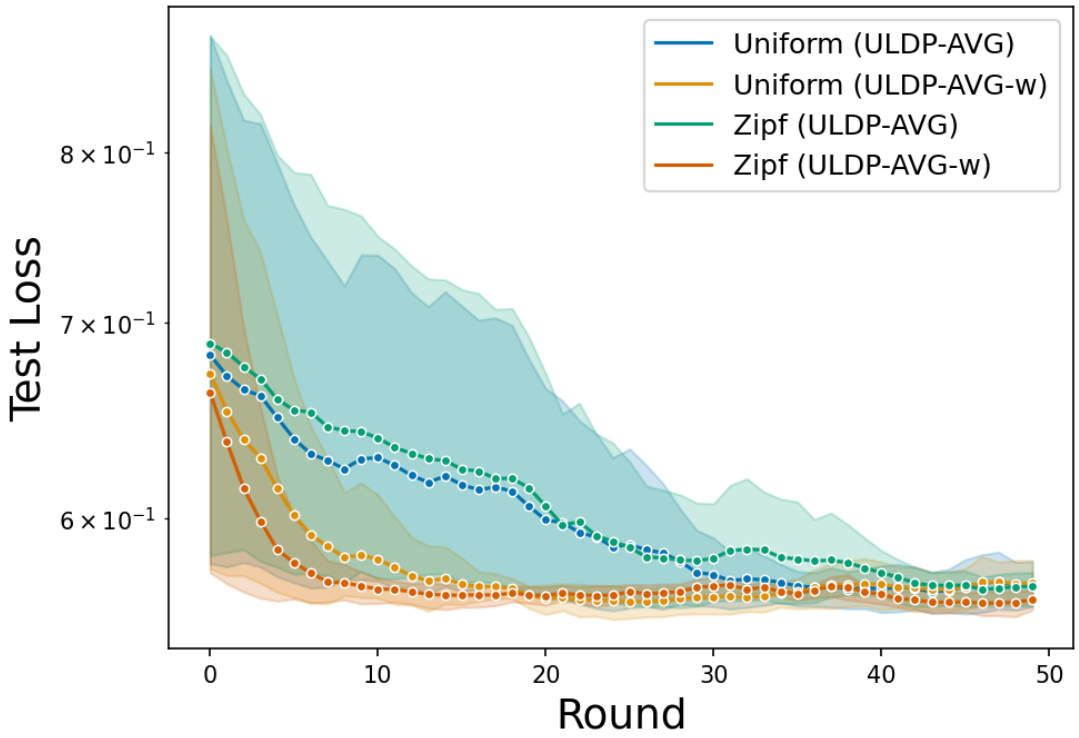


Figure 3.11: Test loss of TcgaBrca: the better weighting is effective especially in biased distribution.

user records, as in the Zipf distribution, giving equal weights (i.e., ULDP-AVG) results in inefficiency and opens up a large gap from ULDP-AVG-w. This trend becomes even more significant as $|S|$ increases because all weights become smaller in ULDP-AVG.

Similarly, Figure 3.11 shows the effect of the better weighting strategy (i.e., ULDP-AVG vs ULDP-AVG-w) on the TcgaBrca dataset. Both uniform and

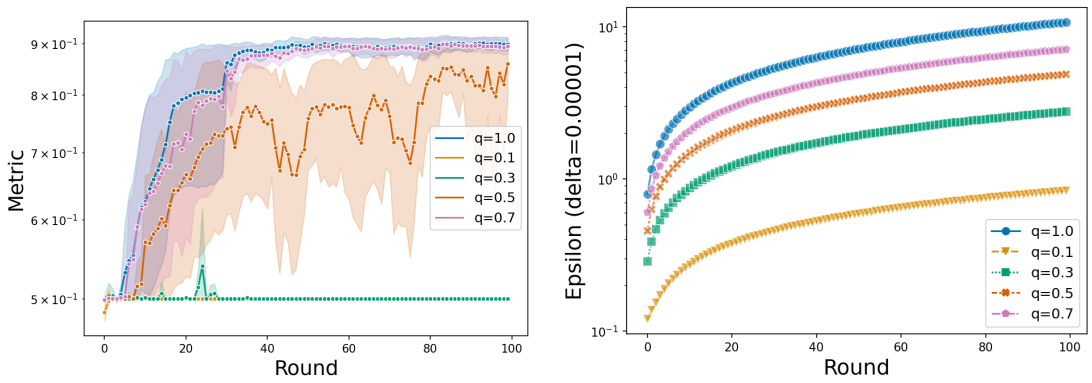


Figure 3.12: In Creditcard, user-level sub-sampling achieves a more competitive privacy-utility trade-off.

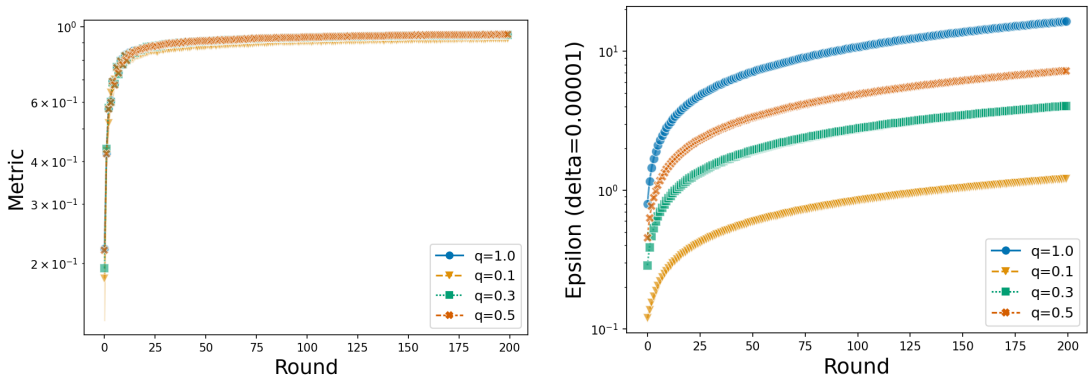


Figure 3.13: In MNIST, user-level sub-sampling achieves a more competitive privacy-utility trade-off.

Zipf distributions have greatly improved in convergence speed with the better weighting strategy. In particular, since the better weighting is adapted to the distribution of the records, it also shows little variance in the results.

User-level sub-sampling effect. We evaluate the effect of user-level sub-sampling. The details of the algorithm of ULDP-AVG with user-level sub-sampling are shown in Algorithm 7. The differences from ULDP-AVG are highlighted in red letters. Figure 3.12 illustrates how user-level sub-sampling affects the privacy-utility trade-offs on the Creditcard dataset with 1000 users. We report the test accuracy and ULDP privacy bounds for various sampling rates $q = 0.1, 0.3, 0.5, 0.7, 1.0$. Basically, a tighter privacy bound is obtained at the expense of utility. As the results show, the degradation of utility due to sub-

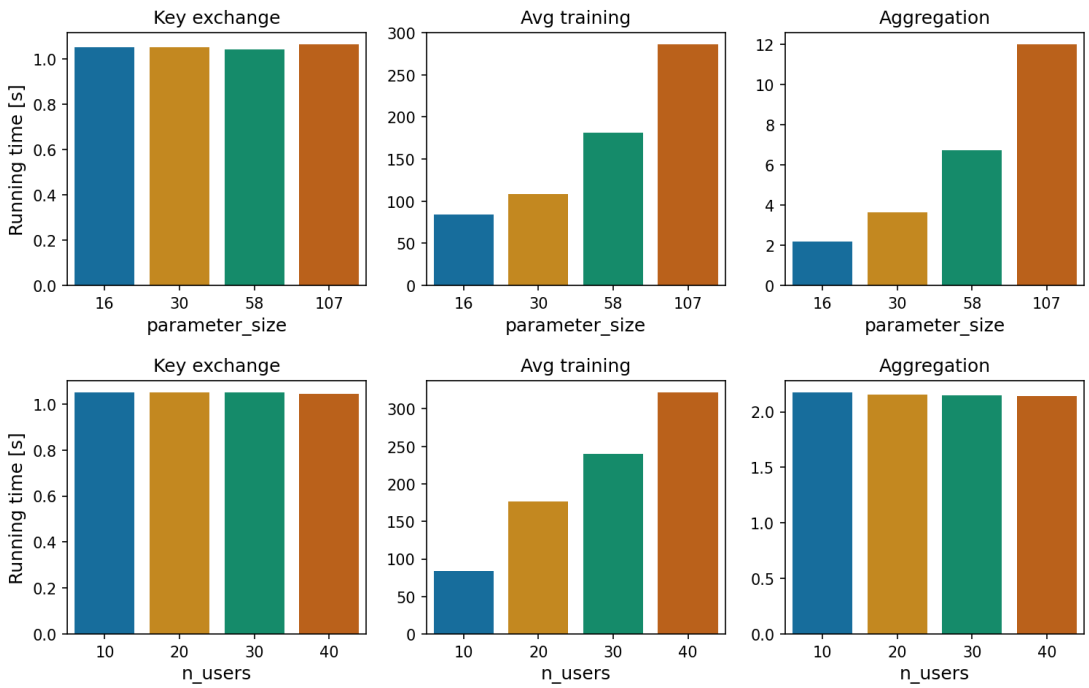


Figure 3.14: The dominant execution time of training grows linearly with parameter size and/or the number of users.

sampling could be acceptable to some extent (e.g., $q = 0.7$) and there could be an optimal point for each setting. Figure 3.13 illustrates how user-level sub-sampling affects the privacy-utility trade-offs on MNIST with 10000 users, with sampling rates $q = 0.1, 0.3, 0.5, 1.0$. The results show that while privacy is greatly improved, there is less degradation in utility. This is due to the fact that there are a sufficient number of users, i.e., 10000. In the case of a larger user base, the effect of sub-sampling is even greater and is likely to become an important technique.

Overhead of private weighting protocol. We evaluate the overheads of execution time with the private weighting protocol. Figure 3.15 shows the execution times following HeartDisease and TcgaBrca with the number of users 10 and 100, respectively, with a skewed (zipf) distribution. These two benchmark scenarios of cross-silo FL from [106] use small models. The left figure shows the time required for local training in each silo, and the right figure shows the execution time for key exchange, preparation of blinded histograms, and aggregation. As shown in the figure, the execution time of local training is dominant and it increases with

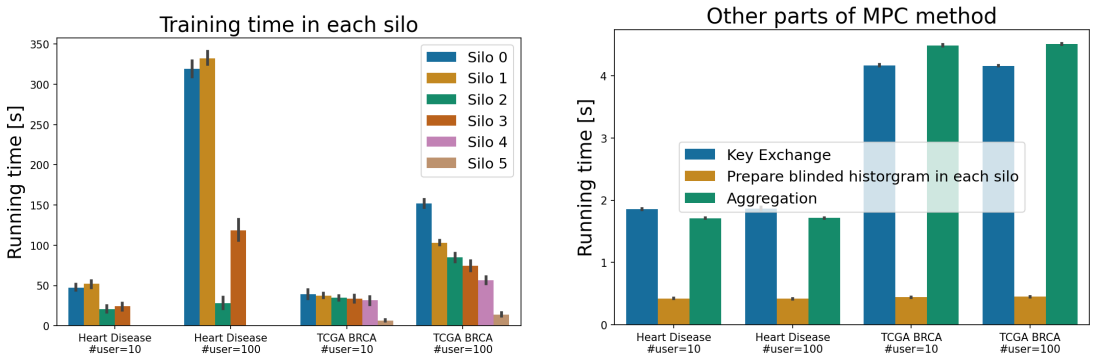


Figure 3.15: With a small model, the private weighting protocol has a practical execution time.

a larger number of users. Overall, it shows realistic execution times under these benchmark scenarios [106]. However, there is still non-negligible overhead, with room for improvement in efficiency.

Figure 3.14 describes the execution times of our proposed private weighting protocol (Protocol 1) with an artificial dataset with 10000 samples and a model with 16 parameters, 20 users, and 3 silos as default. Note that these are on a considerably small scale. The major time-consuming parts of the protocol are key exchange, training in each silo, and aggregation on the server. The upper figure shows the execution times on each part of the protocol with various parameter sizes from 16 to 107 and the bottom figure shows the various number of users from 10 to 40. The execution time of local training is averaged by silos. The dominant part is the local training part, which is considered to be an overhead due to the computation with the Paillier encryption, which grows linearly with parameter size and/or the number of users. The larger parameter size increases the aggregation time on the server as well. Our implementation is based on the Python 3.9 library [d] with 3072-bit security, which itself could be made faster by software implementation or hardware accelerators [159]. However, it can be challenging to apply to larger models, such as DNNs, because the execution time increases linearly on a non-negligible scale with parameter size. Therefore, extending our proposed method to deep models with millions of parameters is a future challenge. It may be possible to replace such software-based encryption methods by using hardware-assisted Trusted Execution Environments, which have recently

[d] `https://github.com/data61/python-paillier`

attracted attention in the FL field [73, 24].

3.6 Conclusion

This study focused on the weaknesses of the privacy of FL and aimed to integrate user-level DP into FL, providing practical privacy guarantees for the trained model in general cross-silo scenarios. We proposed the first cross-silo user-level DP FL framework where a user may have multiple records across silos and designed an algorithm using per-user clipping to directly satisfy ULDP instead of group-privacy. In addition, we developed a better weighting strategy that improves the utility of our proposed method and a novel protocol that performs it privately. Finally, we demonstrated the effectiveness of our proposed method on several real-world datasets and showed that it performs significantly better than existing methods. We also verified that our proposed private protocol works in realistic time in existing cross-silo FL benchmark scenarios.

3.6.1 Future works

- In terms of the privacy aspects, our scenario and privacy definition are fairly generalized based on real-world use cases, but in fact, they can be generalized further. One interesting direction is to consider the case where individual users have their own privacy preferences, i.e., different privacy budgets. This is known as personalized DP [160] and is still an unexplored issue in the cross-silo FL setting and across silo user-level DP.
- Since our proposed method, ULDP-AVG, causes a privacy-utility trade-off, considering ways to improve it can be a future direction. Exploring various strategies to enhance the balance between privacy preservation and utility of the model will be crucial for broader adoption and effectiveness.
- Similarly, the proposed MPC (Multi-Party Computation) protocol generates a trade-off between privacy and efficiency. Our method needs to be further improved, as considerable impracticality was observed for the scalability of the model. Addressing these challenges to enhance the practicality and scalability of our approach will be a significant focus in future work.

CHAPTER 4

OLIVE: Oblivious Federated Learning on Trusted Execution Environment against the risk of sparsification

In this chapter, our focus is on FL with server-side TEE, which is primarily advantageous for privacy and security reasons against an untrusted aggregation server, i.e., **A** and **D** as shown in Table 1.2.

4.1 Introduction

As described in Section 2.3.1, the utilization of server-side TEE has attracted attention. It is advantageous from several perspectives compared to the alternatives such as secure aggregation (SA)[a]. However, pairwise-masking-based SA [66, 97, 96, 161] sacrifices usability in several aspects. It requires time-consuming synchronous distributed mask generation among multiple clients and lacks robustness with respect to participant asynchronicity/dropouts [67], which is difficult to handle and can impede implementation by general practitioners. Further, SA

[a]The recent paper [67] categorized TEE as a method of secure aggregation in FL.

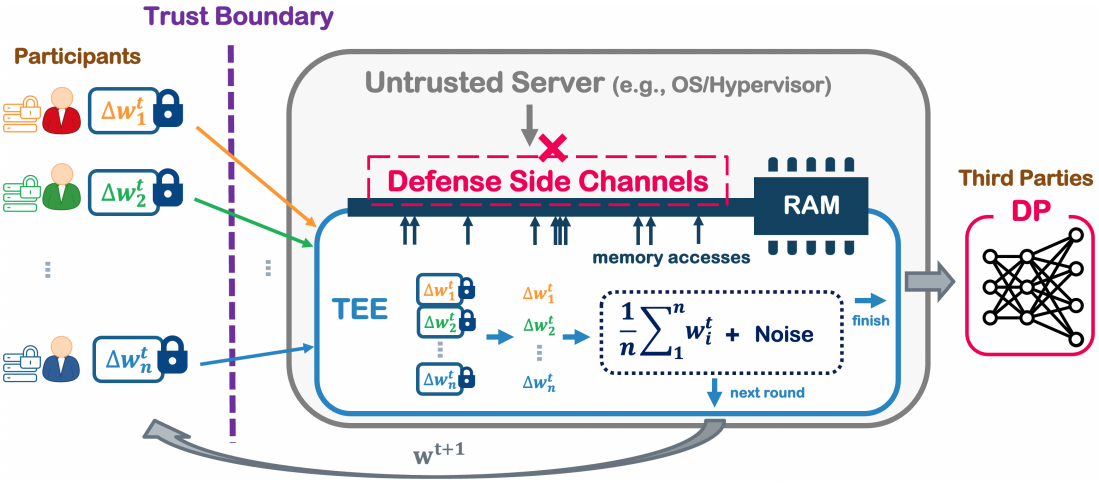


Figure 4.1: OLIVE, i.e., **O**b**LIV**ious f**E**derated learning on TEE is the first method of its kind to prevent privacy risks caused by the leakage of memory access patterns during aggregation in FL rigorously. This allows, for example, to enjoy utility of CDP-FL without requiring a trusted server like LDP-FL.

is inflexible and makes it hard to do extensions, such as Byzantine resistance [71] and asynchrony [75]. In addition, application of gradient sparsification to FL with SA requires either random sparsification [97] or a common sparsified index among multiple clients [161] because of the pairwise constraints, impairing training quality. One simple and important solution to these problems is the use of TEE, even though it requires additional special hardware. In addition to the confidentiality of gradients (i.e., SA functionality), TEE provides remote program integrity and verifiability via *Remote Attestation*, which prevents a malicious server from tampering with the supposed process and provides security against them.

Moreover, FL with TEE addresses the utility gap of differentially private FL (DP-FL) [58, 54, 57] as shown in Table 2.1. The recently studied Shuffle DP-FL [122, 123, 57], which aims to combine the best LDP-FL trust model [56, 119] with the model utility of the CDP-FL [58, 54, 118], exhibits a gap with respect to CDP-FL in terms of utility [57]. As depicted in Figure 4.1, TEE facilitates secure model aggregation on an untrusted server, which ensures only differentially private models are observable by the server. Without trust in the server, as in LDP-FL, model utility is equivalent to that of conventional CDP-FL because any DP mechanism can be implemented within the TEE, whereas the mechanism is restricted when using SA [96]. This important use case, i.e., the combination of

the proposed method with CDP-FL, is analyzed in detail in Section 4.6.

However, implementing a server-side TEE to achieve the aforementioned benefits requires careful analysis of the vulnerabilities of TEE. Several serious vulnerabilities are known to affect TEE owing to side-channel attacks [84, 85, 86], which can cause privacy leakage despite encryption. In particular, such attacks can expose data-dependent memory access patterns of confidential execution and enable attackers to steal sensitive information, such as RSA private keys and genome information [136]. The specific information that may be stolen from these memory access patterns is domain-specific and is not yet known for FL, although several studies have attempted to use TEE for FL [73, 74, 138, 72, 139]. Thus, the extent of the threat of side-channel attacks against FL on a TEE and the types of possible attacks remain critical open problems in this context.

Oblivious algorithms [87, 162, 140] are important leakage prevention techniques that generate only data-independent memory access patterns. A general approach involves making the RAM oblivious, e.g., oblivious RAM (ORAM). PathORAM [162] is known to be the most efficient technique. However, it assumes a private memory space of a certain size and is not applicable to practical TEE, such as Intel SGX [70]. Although Zerotrace [163] addresses this issue, it still incurs significant overhead. Therefore, the design of an algorithm-specific method to obtain an efficient algorithm is an important problem. In this context, [140] proposed an efficient oblivious algorithm for specific ML algorithms, and [164] studied SQL processing. However, an efficient method for FL-specific aggregation algorithm, which can be a vulnerable component of FL with a server-side TEE, has not yet been proposed.

In this study, we address the aforementioned gaps; (1) we clarify privacy risks by designing specific attacks on FL with a server-side TEE and demonstrate them in a real-world scenario; (2) we devise a novel defense against the risks by designing efficient oblivious algorithms and evaluate them empirically on a practical scale. Our analysis reveals that parameter position information is leaked during the execution of the FL aggregation algorithm in a *sparsified* environment. *Sparsification* is often used in FL [90, 91, 97, 161] to reduce communication costs and/or improve model accuracy [165]. The goal of an attacker is to infer a set of sensitive labels included in the target user's training data, similar to the goal described in [23, 166]. We assume the attacker to be capable of observing memory access patterns, accessing the dataset that covers the overall dataset distribu-

tion, and accessing the model trained during each round. Although sparsified index information in FL has been considered as somewhat private information in previous studies [161, 122], unlike in our study, no specific attacks have been investigated. After demonstrating the proposed attack on real-world datasets, we propose efficient oblivious algorithms to prevent such attacks completely. To this end, we carefully construct existing oblivious building blocks, such as the oblivious sort [167] and our designed components. Our proposed method OLIVE, an **O**b**LIV**ious f**E**derated learning system based on server-side TEE, is resistant to side-channel attacks, enabling truly privacy-preserving FL. In addition to fully oblivious algorithms, we further investigate optimization by adjusting the data size in the enclave, and study more efficient algorithms by relaxing the definition of obliviousness. Finally, we conduct extensive experiments on real-world data to demonstrate that the proposed algorithm, designed for FL aggregation, is more efficient than the general-purpose PathORAM with SGX [163].

The contributions of this study are summarized below:

- We analyze the exposure of memory access patterns to untrusted servers when TEE is used for model aggregation in FL. A risk is identified in the context of sparsified gradients, which are often used in recent FL.
- We design a supervised learning-based sensitive label inference attack based on index information observed from side-channels of sparsified gradients. We demonstrate the attack on a real-world dataset. One of the results reveals that when training with a CNN on CIFAR100 with top-1.25% sparsification, the sensitive labels of training data (each participant is assigned 2 out of 100 labels) are leaked with approximately 90% or better accuracy (Figure 4.7).
- We propose a novel oblivious algorithm that executes model aggregation efficiently by combining oblivious primitives, such as oblivious sort and certain designed components. The efficiency of the proposed method is verified via extensive experiments. In particular, it is demonstrated to be more than 10 $\times$ faster than a PathORAM-based method and require only a few seconds even in cases involving a million parameters (Figure 4.11).

The remainder of this chapter is organized as follows. Preliminary notions are presented in Section 4.2. The overview of the proposed system and the problem

setting is described in Section 4.3. Sections 4.4 and 4.5 demonstrate the proposed attack and defense, respectively, with empirical evaluations. Section 4.6 shows the relationship between Olive and CDP-FL. Section 4.7 discusses related works and Section 4.8 concludes.

4.2 Preliminaries

4.2.1 Federated Learning with Sparsification

To reduce communication costs and improve model accuracy, a sparsification of the model parameters before their transmission to the server has been extensively studied in FL [90, 91, 97, 161, 103, 168, 92]. All of the aforementioned methods sparsify parameters on the client side, apply an encoding that represents them as *value* and *index* information [168], transmit them to the server, and aggregate them into a dense global model on the server side. Exceptionally, [103, 161] used common sparsification among all clients using common sparsified indices and aggregated them into a sparse global model. However, as observed in [97], there is practically little overlap among the top-k indices for each client in real-world data, especially in the non-i.i.d. environment, which is common in FL. This highlights one of the limitations of pairwise masking-based SA [97, 161] (see Section 4.7). In general, top-k sparsification is the standard method. By transmitting only the top-k parameters with large absolute gradients to the aggregation server, communication cost is reduced by more than 1 to 3 orders of magnitude [91]. In terms of model utility, this technique outperforms the random selection of k indices (random-k) [97], particularly when the compression ratio is smaller than 1% [91, 168, 103, 161]. Other sparsification methods, such as threshold-based [91], top-k under LDP [120] and the recently proposed convolutional kernel [168], also exist. However, these sparsified gradients can lead to privacy leakages through the index. In [161, 122], the set of user-specific top-k indices was treated as private information; however, no specific attacks were investigated.

4.2.2 Memory Access Pattern Leakage of TEE

As we introduced in Section 2.3, although the data are encrypted and cannot be viewed in enclaves, observable memory/page access patterns from side-channels

can reveal private information [84, 85, 136, 137, 86]. To prevent such attacks, oblivious algorithms have been proposed to hide access patterns during the secure execution of the process. An oblivious algorithm is defined as follows.

Definition 8 (Oblivious algorithm [169])**.** *An algorithm $\mathcal{M}$ is δ-statistically oblivious if, for any two input data I and I' of equal length and any security parameter λ, the following relation holds:*

$$\mathbf{Accesses}^{\mathcal{M}}(\lambda, I) \overset{\delta(\lambda)}{\equiv} \mathbf{Accesses}^{\mathcal{M}}(\lambda, I')$$

where $\mathbf{Accesses}^{\mathcal{M}}(\lambda, I)$ denotes a random variable representing the ordered sequence of memory accesses. The algorithm $\mathcal{M}$ is generated upon receiving the inputs, λ and I. $\overset{\delta(\lambda)}{\equiv}$ indicates that the statistical distance between the two distributions is at most $\delta(\lambda)$. The term δ is a function of λ which corresponds to a cryptographic security parameter. When δ is negligible, we say that $\mathcal{M}$ is fully oblivious, and when δ is 1, it is not oblivious.

A typical approach for constructing an oblivious algorithm utilizes an ORAM, such as PathORAM [162]. Although ORAMs are designed for general use as key-value stores, several oblivious task-specific algorithms, such as ML [140] and SQL processing [164] (see Section 4.7 for details), have been proposed from a performance perspective. They are constructed based on oblivious sort [167] and/or access to all memory (i.e., linear scan), and are distinct from ORAM at the algorithmic level. Further, ORAM generally assumes that the existence of a trusted memory space such as *client storage* [162], which is incompatible with the SGX assumption of leaking access patterns in enclaves. Thus, only CPU registers should be considered to be trusted memory spaces [163]. [140] implemented oblivious ML algorithms using `CMOV`, which is an x86 instruction providing a conditional copy in the CPU registers. `CMOV` moves data from register to register based on a condition flag in the register, which is not observed by any memory access patterns. Using the `CMOV` instruction, conditional branching can be implemented with a constant memory access pattern that does not depend on the input, thereby removing the leakage of subsequent code addresses. For example, Zerotrace [163] implements PathORAM on SGX by obliviously implementing client storage based on `CMOV`. We can construct and use low-level oblivious primitives, such as *oblivious move* (`o_mov`, Listing 4.1) and *oblivious swap* (`o_swap`, Listing 4.2). `o_mov(flag,x,y)`

```
int o_mov(bool flag, uint64 x, uint64 y) {
    /* inline assembly */
    /* register mapping:
        flag  => ecx, x  => rdx, y  => r8  */
    mov rax, rdx
    test ecx, -1
    cmovz rax, r8
    return rax
}
```

Listing 4.1: Oblivious move based on `CMOV`

```
int o_swap(bool flag, uint64 x, uint64 y) {
    /* inline assembly */
    /* register mapping:
        flag  => rax, x  => rdx, y  => r8  */
    test rax, rax
    mov r10, r8
    mov r9, rdx
    mov r11, r9
    cmovnz r9, r10
    cmovnz r10, r11
    mov rdx r9
    mov r8, r10
}
```

Listing 4.2: Oblivious swap based on `CMOV`

is a function that accepts a Boolean condition flag as its first argument and returns `x` or `y` depending on the flag. Therefore, designing an appropriate oblivious algorithm for SGX requires a combination of high-level algorithm designs, such as the oblivious sort and low-level primitives.

We describe the detailed implementation of the oblivious primitive used in Listing 4.1 and 4.2. The C inline assembler-like pseudo-code is shown here. However, the Rust implementation we actually used is available in the public repository.

4.3 Proposed System

In this section, we first clarify our scenario and threat model, and then present a system overview of the Olive. Finally, we analyze the details of the potential privacy risk, followed by discussion of a specific privacy attack and evaluation in Section 4.4.

4.3.1 Scenario

We target a typical FL scenario with a single server and clients using identical format data (i.e., horizontal FL). The server is responsible for training orchestration, aggregating parameters, updating the global model, selecting clients for each training round, and validating model quality. The server-side machine is assumed to be placed in a public or private environment [72, 76] and is equipped with a TEE capable of RA (e.g., Intel SGX).

Threat model. We assume an adversary to be a *semi-honest* server that allows FL algorithms to run as intended, while trying to infer the sensitive information of clients based on shared parameters. This is a compatible threat model with those in existing studies on FL with SA [66] and even with server-side TEE [72, 73, 74]. The semi-honest threat model is selected despite using TEE, because the assumed attack in this work does not diverge from the established FL protocol. The goal of the adversary is not to damage the availability (e.g., DoS attacks) or undermine the utility of the model (e.g., data-poisoning attacks) [66, 71, 170] as *malicious* attackers in FL context. Note that several side-channel attacks against TEE require malicious (i.e., privileged) system software, which we distinguish from an attacker and categorize as *malicious* in FL. Nevertheless, [171] reported that malicious servers improve inference attacks in FL. In Section 4.5.6, we discuss the relationship between such malicious servers and the privacy and security of the proposed system.

We assume that the server has (1) access to the trained model during each round of FL, (2) access to the global test dataset, and (3) the capability to observe the memory access patterns of the TEE. These requirements can be justified as follows. (1): Because the server is in charge of model validation, it makes sense for the server to have access to the global models during all rounds. Alternatively, attackers can easily blend in with clients to access global models. (2): Generally,

the semi-honest server that has access to public datasets for model validation covers the overall dataset distribution, which is essential in production uses. Similar assumptions have been made in previous studies on inference attacks [172, 103]. Subsequently, we experimentally evaluate the required dataset volume (Figure 4.9). (3): This follows the general threat assumption for TEE. The SGX excludes side-channel attacks from the scope of protection [70, 86]. Except for the trusted hardware component (i.e., the CPU package), all other components of the server, e.g., the system software (i.e., OS/hypervisor), main memory, and all communication paths, are considered to be untrusted. The server can observe memory access patterns through known or unknown side-channel attacks, as described in Section 4.2.2.

4.3.2 System Overview

The proposed system, namely the Olive (Figure 4.1), follows the basic FedAVG algorithm with standard top-k sparsification; however, the TEE is placed on the server side with a server-side algorithm resistant to side-channel attacks. As an initial configuration, we provide an enclave in which each client verifies the integrity of the processes running on the enclave via RA and exchanges shared keys (AES-GCM). If attestation fails, the client must refuse to join the FL in this phase. We assume that communication between the client and server is performed over a secure channel (TLS), which the untrusted server terminates, and that the transmitted gradients[b] are doubly encrypted and can only be decrypted in the trusted enclave.

The overall algorithm of the Olive is presented in Algorithm 9, where the differences with respect to the basic FedAVG algorithm are highlighted in red. The initial provisioning is omitted and a different shared key, sk_i, is stored in the enclave for each user, i ($\in [N]$) (Line 1). In each round, the participants are securely sampled in the enclave (Line 4). The selected users are memorized in the enclave and used for client verification (Line 9) after the encrypted data are loaded into the enclave (Line 8). On the client side, locally trained parameters are top-k sparsified (Line 21), and then encoded and encrypted (Line 22). The

[b]In FedAVG, the data shared by users are not exactly gradients—rather, they are the delta of model weights. However, in the context of compatibility with FedSGD, we jointly refer to model update data transmitted by users as *gradients* or *parameters*.

encrypted data loaded into the enclave are decrypted and verified (Line 11). Verification (Lines 9, 11) is not essential to our work; however, it prevents man-in-the-middle attacks and biased client selection. As discussed in Section 4.3.3, the aggregation operation (Line 12) is required to be oblivious, and we present lower-level and detailed algorithms in Section 4.5 to this end. In accordance with the principle that the Trusted Computing Base (TCB) should be minimized, only the aggregation operation is performed in the enclave. Finally, the aggregated parameters are loaded outward from the enclave (Line 13). Thus, the parameters transmitted by all clients remain completely invisible to the server—only the aggregated parameters are observable.

Algorithm 9 OLIVE: Oblivious FL on TEE

Input: N: # participants, η_c, η_s: learning rate

KeyStore $\leftarrow$ *Remote Attestation* with all user i ▷ key-value store in enclave that stores sk_i: user i's shared key from RA in provisioning
procedure TRAIN(q, η_c, η_s)
  Initialize model θ^0
  for each round $t = 0, 1, 2, \ldots$ **do**
    $\mathcal{Q}^t \leftarrow$ (sample users from N for round t) ▷ securely in enclave
    for each user $i \in \mathcal{Q}^t$ **in parallel do**
      $\text{Enc}(\Delta_i^t) \leftarrow \text{ENCCLIENT}(i, \theta^t, \eta_c)$
      LoadToEnclave($\text{Enc}(\Delta_i^t)$)
      check if user i is in $\mathcal{Q}^t$
      $sk_i \leftarrow$ KeyStore[i] ▷ retrieve user i's shared key
      $\Delta_i^t \leftarrow \text{Decrypt}(Enc(\Delta_i^t), sk_i)$
    end for
    /* **Obliviously performed, such as Algorithm 12 or 13** */
    $\tilde{\mathbf{\Delta}}^{\mathbf{t}} = \frac{1}{|\mathcal{Q}^t|} \sum_{\mathbf{i} \in \mathcal{Q}^t} \mathbf{\Delta_i^t}$ ▷ **oblivious algorithm**
    LoadFromEnclave($\tilde{\Delta}^t$)
    $\theta^{t+1} \leftarrow \theta^t + \eta_s \bar{\Delta}^t$
  end for
end procedure
procedure ENCCLIENT(i, θ^t, η, C)
  $\theta \leftarrow \theta^t$
  $\mathcal{G} \leftarrow$ (user i's local data split into batches)
  for batch $g \in \mathcal{G}$ **do**
    $\theta \leftarrow \theta - \eta \nabla \ell(\theta; g)$
  end for
  $\Delta \leftarrow \theta - \theta^t$
  $\Delta \leftarrow \text{TopkSparse}(\Delta)$ ▷ top-k sparsification on gradients
  $\text{Enc}(\Delta') \leftarrow \text{Encrypt}(\Delta, sk_i)$ ▷ Authenticated Encryption (AE) mode, such as AES-GCM, with shared key, sk_i, from RA
  return Enc(Δ)
end procedure

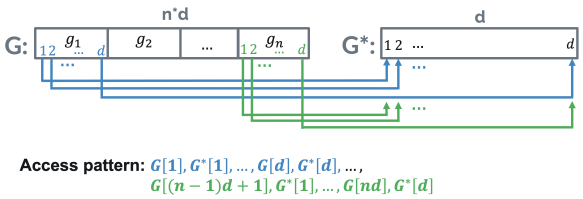


Figure 4.2: Dense gradients induce uniform access patterns.

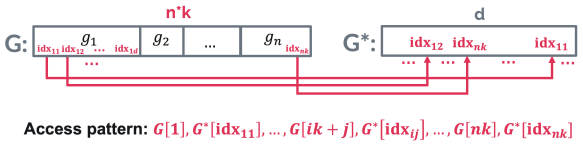


Figure 4.3: Sparse gradients induce biased access patterns.

4.3.3 Security Analysis

Although TEE enables model training while protecting raw gradients, an untrusted server can observe the memory access patterns, as described in Section 4.2.2. Here, we analyze the threats that exist based on memory access patterns.

For formal modeling, let g_i denote the k-dimensional gradient transmitted by user i and let g^* be the d-dimensional global parameter after aggregation. In the typical case, $k = d$, when dense gradients are used. Let $\mathbf{G}_i$ and $\mathbf{G}^*$ denote the memories required to store the gradients of g_i and g^*, respectively, and let the number of clients participating in each round be n. The memory that stores the entire gradient is denoted by $\mathbf{G} = \mathbf{G}_1 \parallel ... \parallel \mathbf{G}_n$, where $\parallel$ denotes concatenation. A memory access, a, is represented as a triple $a = (\mathbf{A}[i], \texttt{op}, \texttt{val})$, where $\mathbf{A}[i]$ denotes the i-th address of the memory, $\mathbf{A}$; `op` denotes the operation for the memory—either `read` or `write`; and `val` denotes the value to be written when `op` is `write`, and `null` otherwise. Therefore, the observed memory access pattern, **Accesses**, can be represented as **Accesses** $= [a_1, a_2, ..., a_m]$ when the length of the memory access sequence is m.

In FL, operations performed on the server side generally consist of summing and averaging the gradients obtained from all users. We first note that this procedure is oblivious to *dense gradients*. As depicted in Figure 4.2, the summing operation involves updating the value of the corresponding index of $\mathbf{G}^*$ while performing a linear scan on $\mathbf{G}$, where memory accesses are performed in a fixed order and at fixed addresses, irrespective of the content of $\mathbf{G}$. We refer to this general summing part as the *linear* algorithm and present it in Algorithm 10 for completeness. (The

Algorithm 10 Linear algorithm (and averaging and perturbing)

Input: $\mathbf{G} = \mathbf{G}_1 \parallel ... \parallel \mathbf{G}_n$ where $\mathbf{G}_p$ ($p \in [n]$) is gradient from user p and k length vector, $\mathbf{G}$ is nk length vector and $\mathbf{G}$'s element g_q ($q \in [nk]$) is composed of (`index`, `value`)

Output: $\mathbf{G}^*$: Aggregated gradient and d length vector

```
1: procedure AGGREGATION(G)
2:     /* linear algorithm */
3:     Initialize gradients G*
4:     for i = 1, ..., n do
5:         for j = 1, ..., k do
6:             G*[G[k * (i − 1) + j].index] += G[k * (i − 1) + j].value
7:         end for
8:     end for
9:     /* Averaging and Perturbing with linear access */
10:    for i = 1, ..., d do
11:        G*[i] /= n
12:    end for
13:    for i = 1, ..., d do
14:        z ← Random noise (e.g., Gaussian distribution)
15:        G*[i] += z
16:    end for
17:    return G*
18: end procedure
```

main focus here is on which memory addresses are accessed in the operation.)

Proposition 2. *The linear algorithm is fully oblivious to dense gradients.*

Proof. Let the access pattern of *linear* algorithm for dense gradients be $\mathbf{Accesses}^{\mathbf{dense}}$; then, the pattern is represented as follows:

$$\begin{aligned}&\mathbf{Accesses}^{\mathbf{dense}} = \\ &\quad [(\mathbf{G}[1], \texttt{read}, *), (\mathbf{G}^*[1], \texttt{read}, *), (\mathbf{G}^*[1], \texttt{write}, *), ..., \\ &\quad (\mathbf{G}[nd], \texttt{read}, *), (\mathbf{G}^*[d], \texttt{read}, *), (\mathbf{G}^*[d], \texttt{write}, *)]\end{aligned}$$

This means reading the sent gradients $\mathbf{G}[id+j]$, reading the corresponding aggregated gradients $\mathbf{G}^*[j]$, adding them together, and then writing them to aggregated gradient $\mathbf{G}^*[j]$ again, for any $i \in [n]$ and $j \in [d]$. For any two input data I, I'

of equal length, for any security parameter λ, $\mathbf{Accesses}^{\mathbf{dense}}$ is identical and the statistical distance $\delta = 0$. Finally, *linear* algorithm is 0-statistical oblivious. □

The linear algorithm is executed in $O(nd)$ because all the elements of the gradient $\mathbf{G}$ are accessed. In addition, the averaging operation only accesses $\mathbf{G}^*$ linearly in $O(d)$, which is obviously fully oblivious.

However, when the gradients are *sparsified*, which is often an important scenario in FL, the access pattern of the *linear* algorithm is not oblivious, and sensitive information may be leaked. The weights of sparse gradients are generally given by tuples of `index`, which hold the location information of the parameter, and a `value`, which holds the gradient value. This is irrespective of its quantization and/or encoding because it requires calculating the sum of the original dense gradients. Figure 4.3 depicts the access pattern when an aggregation operation is used for sparsified gradients.

Proposition 3. *The linear algorithm is **not** oblivious to sparsified gradients.*

Proof. Linear access to $\mathbf{G}$ for sparsified gradients occurs when the access pattern, $\mathbf{Accesses}^{\mathbf{sparse}}$, satisfies

$$\begin{aligned}&\mathbf{Accesses}^{\mathbf{sparse}} = \\ &\quad[(\mathbf{G}[1], \texttt{read}, *), (\mathbf{G}^*[\mathrm{idx}_{11}], \texttt{read}, *), (\mathbf{G}^*[\mathrm{idx}_{11}], \texttt{write}, *), ..., \\ &\quad(\mathbf{G}[nk], \texttt{read}, *), (\mathbf{G}^*[\mathrm{idx}_{nk}], \texttt{read}, *), (\mathbf{G}^*[\mathrm{idx}_{nk}], \texttt{write}, *)]\end{aligned}$$

where the `index`es of sparsified gradients of user i are $\mathrm{idx}_{i1}, .., \mathrm{idx}_{ik}$ for all $i \in [n]$. The access pattern, $\mathbf{Accesses}^{\mathbf{sparse}}$, is deterministic and corresponds in a one-to-one fashion with the sequence of the `index`es of the input data. Considering two input data, I and I', with different sequences of `index`es, no overlap exists in the output distribution. Then, the statistical distance between them is 1. □

The access pattern on the aggregated gradients, $\mathbf{G}^*$, reveals at least one set of indices $\{\mathrm{idx}_{ij} \,|\, j \in [d]\}$ for each user i, depending on the given gradients. Considering data-dependent sparsifications, such as top-k, which are generally used in FL, the gradient indices of the sparsified gradients may be sensitive to the training data. In the next section, we demonstrate that privacy leakage can be caused on a real-world dataset.

Generality and Limitation. Let us now clarify the format and method of sparsified gradients. Although various quantization and/or encoding methods in

FL have been studied (e.g., [101]), quantization is irrelevant to the problem of leakage considered in this study because it affects only the values and not the index, and encoding is irrelevant because it is eventually decoded on the server side. For example, in [97, 161], the index location information was encoded in d-dimensional one-bit array, but the same problem occurred during aggregation. As aggregation is performed on the original dense gradients, each update requires access to a specific index of the dense gradients ($\mathbf{G}^*$), resulting in identical access patterns. It should also be noted that risk is sparsification-dependent. If the client's training data and observed indices are uncorrelated, then index leakage is not considered to be a risk. For example, when random-k is adopted, as in [97], no risk is involved. While threshold-based sparsification [91] is almost identical to top-k, LDP-guaranteed index [120] and the recently proposed convolution-kernel-based index [168] are still unclear. These index information can correlate to some extent with the client's training data, but not as much as top-k. The scope of our study is limited to the demonstration that attacks are possible with the standard top-k—the investigation of various other sparsifications are left for future research.

4.4 Attack on Gradient index

4.4.1 Design

In this section, we design a server-side attack to demonstrate that privacy leakage of the training data can occur based on the index information in the gradients. We assume a sparsified gradient based on top-k [173, 90, 91]. The attacker is assumed to satisfy the assumptions listed in Section 4.3.1. The proposed attacks can be used to raise awareness of the security/privacy risks of FL on TEE, which have not been reported in related works [139, 73, 138, 72], and also serve as an evaluation framework for defenses.

Algorithm 11 Attack on index: JAC or NN

Input: i: target user, X_l: test data with label l ($l \in L$), round: T

index $\leftarrow \{\}$ ▷ observed access patterns
/* Prepare teacher and target indices */
teacher $\leftarrow \{\}$ ▷ teacher access patterns to train a classifier
for each round $t = 1, \ldots, T$ **do**
    /* T_i: rounds participated in by user i */
    if $t \in T_i$ **then**
        /* $A_i^{(t)}$: observed top-k indices of user i of round t */
        Store $A_i^{(t)}$ to **index**$[i, t]$
        for each label $l \in L$ **do**
            /* θ^t: the global model after round t */
            /* $I_l^{(t)}$: top-k indexes training with θ^t and X_l */
            Store $I_l^{(t)}$ to **teacher**$[l, t]$
        end for
    end if
end for
/* Calculate scores for each label l */
S $\leftarrow []$ ▷ form of [(label, similarity)]
/* If JAC: Jaccard similarity-based scoring (SIM) */
for each label $l \in L$ **do**
    Store (l, SIM($\|_{\tau \in T_i}$**index**$[i, \tau]$, $\|_{\tau \in T_i}$**teacher**$[l, \tau]$) to **S**
end for
/* If NN: neural network-based scoring */
Train the model M_t with **teacher**$[l, t]$ ($l \in L$) for each t ($\in T$)
for each label $l \in L$ **do**
    Store (l, PREDICT($M_1, ..., M_T, \|_{\tau \in T_i}$**index**$[i, \tau]$)) to **S**
end for
/* If NN-SINGLE: using single neural network */
Train the model M_0 with $\|_{\tau \in T}$**teacher**$[l, \tau]$ ($l \in L$)
for each label $l \in L$ **do**
    Store (l, PREDICT($M_0, \|_{\tau \in T}$**index**$[i, \tau]$)) to **S**
end for
/* 1D K-Means clustering KMEANS */
[labels, centroid] $\leftarrow$ KMEANS(**S**)
return labels of the cluster with the largest centroid

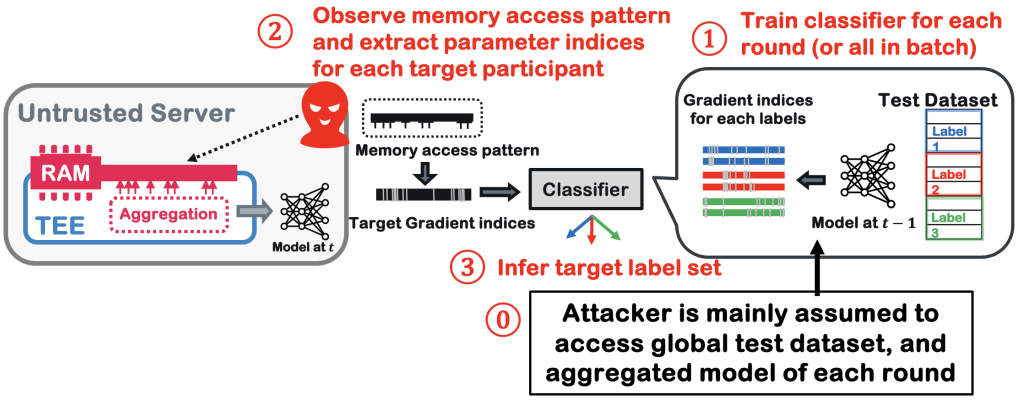


Figure 4.4: Overview of the proposed attack: Attacker is mainly assumed to access global test dataset, and aggregated model of each round, and the attacker trains classifier for each round (or all in batch) that takes parameter indices as input and outputs the target label set, observes memory access pattern through side-channels and extract parameter indices for each target participant and infers target label set.

The goal of the attack is to infer the target client's sensitive label information based on the training data. For example, when training FL on medical image data, such as image data on breast cancer, the label of the cancer is very sensitive, and participants may not want to reveal this information. A similar attack goal was considered in [23, 166]. Our designed attack is based on the intuition that the top-k indices of the locally converged model parameters are correlated with the labels of the local training data. We train a classifier that accepts the observed index information as the input by supervised learning using a public test dataset and the output is the sensitive label set. Access to the dataset is justified, for example, by the need for model validation, as described in Section 4.3.1 and in previous studies on inference attacks [172, 103]. We design two basic methods—the Jaccard similarity-based nearest neighbor approach (Jac) and a neural network (NN). The detailed algorithm is presented in Algorithm 11. An overview of these methods is provided below and illustrated in Figure 4.4.

1. First, the server prepares the test data X_l with label l for all $l \in L$, where L denotes the set of all possible labels.

2. In each round t ($\in T$), an untrusted server observes the memory access patterns through side-channels, obtains the index information of the top-k gradient

indices $\mathbf{index}[i, t]$ for each user i, and stores it (Lines 4–8).

3. The server computes the gradient of the global model with θ^t and X_l, without model updates for each round t ($\in T$), using the test data categorized by labels, and obtains the top-k indices $\mathbf{teacher}[l, t]$ as teacher data for each label (Lines 9–12).

4. After the completion of all rounds T, in JAC, we calculate the Jaccard similarity between observed access patterns, $\|_{\tau \in T_i} \mathbf{index}[i, \tau]$ and $\|_{\tau \in T_i} \mathbf{teacher}[l, \tau]$, for each label l (Lines 15–17). Jaccard similarity is selected because, in the worst-case scenario, the index information transmitted by a participant is randomly shuffled, rendering the sequence meaningless.

5. In NN, the attacker trains neural networks using $\mathbf{teacher}[l, t]$, with indices as the features and labels as the target (Line 19). The outputs of the model are the scores of the label. Subsequently, we use a trained model to predict the labels included in the training data corresponding to the input, $\mathbf{index}[i]$. For this task, we design the two following NN-based methods. In the first method, a model, M_t, is trained during each round, t, and the output scores of the models are averaged to predict the labels (NN). In the second method, a single model, M_0, is trained using the concatenated indices of the entire round as input and a single output is obtained (NN-SINGLE). In the experiment, both cases involve a multilayer perceptron with three layers (described in Appendix 4.9.1 of this chapter). Note that as the model input, index information is represented as a multi-hot vector. In the case of NN-SINGLE, each client participates in only a proportion of the rounds—the indices of the rounds they do not participate in are set to zero as the input to the model. Although NN-SINGLE is expected to be able to capture the correlation over rounds better than NN, this zeroization may reduce the accuracy. Finally, as in JAC, we store the scores for each label obtained via model prediction (Lines 20–21).

6. If the number of labels of the target client is known, the scores are sorted in descending order and the highest labels are returned. If the number of labels is unknown, K-means clustering is applied to the scores to classify them into 2 classes, and the labels with the highest centroid are returned (Lines 23–24).

Finally, the information obtained from the side-channels can also be used to design attacks for other purposes, such as additional features in reconstruction

[174] or other inference attacks [175]. The aim of this study is simply to demonstrate that the top-k gradient indices that can be observed on untrusted servers contain sufficient information to cause privacy leakages; therefore, we leave the study of attacks for different purposes to future research.

4.4.2 Evaluation Task

In our evaluation of attacks, the server performs an inference attack on any client in the scenario detailed in Section 4.3.1. The clients have a subset of labels, and the attacker's goal is to infer the sensitive label set of a target client based on their training data. The attacker selects any subset or the entire set of users and performs an inference attack on each user. We utilize *all* and *top-1* as accuracy metrics for evaluating attack performance. We define *all* as the percentage of clients that match the inferred labels exactly, e.g., the inferred label set is 1,3,5, and the target client's label set is 1,3,5. We define *top-1* as the percentage of clients that contain the highest scored inferred label, e.g., the highest scored inferred label is five, and the target client's label set is 4,5, which we consider to be a minimal privacy leak. In addition, we adjust the distribution of the label set such that the client is able to control the difficulty of the attack. The number of labels in the set and the number of labels that are *fixed* or *random* are configurable. In the case of a *fixed* label, all users exhibit the same number of labels, which is known to the attacker. In the case of the *random* label, the maximum number is assigned, and all users exhibit various numbers of labels. Generally, *random* label and larger numbers of labels are more difficult to infer.

4.4.3 Empirical Analysis

Here, we demonstrate the effectiveness of the designed attack.

Setup. Table 4.1 lists the datasets and global models used in the experiments. Details of the model, including the attacker's NN, are provided in Appendix 4.9.1 of this chapter. In addition to the well-known image datasets, MNIST and CIFAR 10 and 100, we also use Purchase100, which comprises tabular data used in [52] for membership inference attacks. We train the global models using different numbers of parameters, as listed in Table 4.1. The learning algorithm is based on Algorithm 9, in which we provide the sparse ratio, α, instead of k in top-k. FL's learning parameters include the number of users, N; the participant

Table 4.1: Datasets and global models in the experiments.

Dataset	Model (#Params)	#Label	#Record (Test)
MNIST	MLP (50890)	10	70000 (10000)
CIFAR10	MLP (197320) CNN (62006)	10	60000 (10000)
Purchase100	MLP (44964)	100	144000 (24000)
CIFAR100	CNN (201588)	100	60000 (10000)

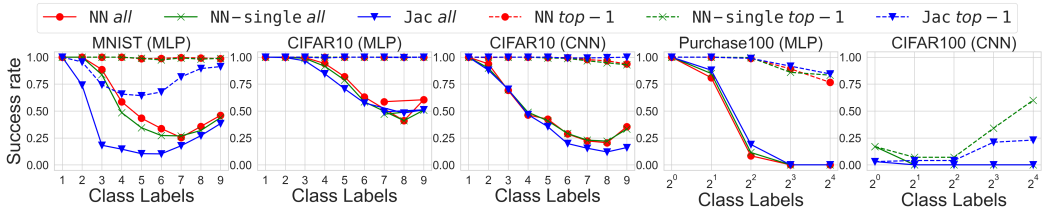


Figure 4.5: Attack results on datasets with a fixed number of labels: Vulnerable, especially when there are few labels.

sampling rate, q; the number of rounds, T. The default values are given by $(N, q, T, \alpha) = (1000, 0.1, 3, 0.1)$. The attack methods are evaluated for JAC, NN, and NN-SINGLE, as described in the previous section. T is smaller than that in normal FL scenarios, which implies that our method requires only a few rounds of attacks. All experimental source codes and datasets are open[c].

Results. Figure 4.5 depicts the attack results for NN, NN-SINGLE, and JAC on all datasets with a *fixed* number of labels, and Figure 4.6 presents the results with a *random* number of labels. In CIFAR100, $T = 1$ is used because the model size is large. The y-axis represents the success rate of the attacks, and the x-axis represents the number of labels possessed by each client. When the number of labels is small, all three attacks exhibit a high probability of success. The success rate of *top-1* is high irrespective of the number of labels, whereas *all* decreases with each additional label. On CIFAR10, the MLP model maintains a higher success rate for a large number of labels compared to the CNN model. This indicates that the complexity of the target model is directly related to the contribution of the index information to the attack. The NN-based method is more powerful on

[c] `https://github.com/FumiyukiKato/FL-TEE`

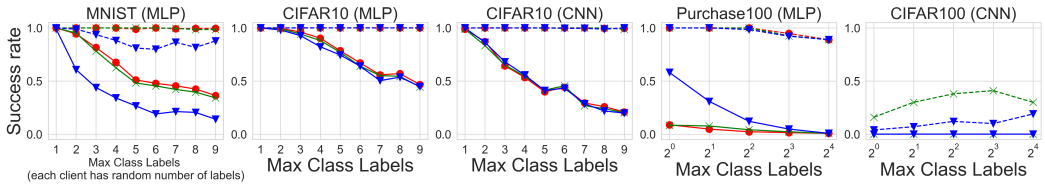


Figure 4.6: Attack results on datasets with a random number of labels (more difficult setting): When the number of labels is low, the attacker can attack the client without knowing the exact number of labels.

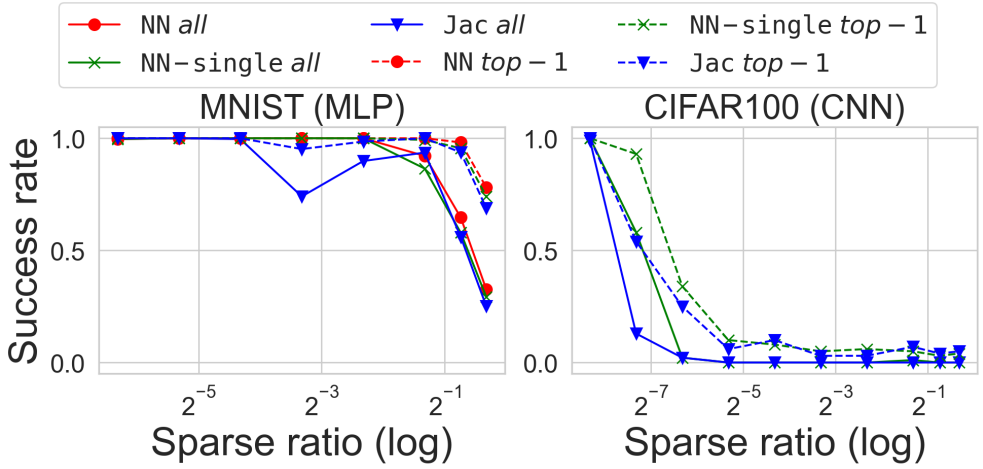


Figure 4.7: Attack results w.r.t. sparse ratios: Higher the sparsity, the more successful the attack tends to be.

MNIST, but it performs similarly to the other methods on the other datasets. This indicates that the gradient index information is not complex and can be attacked using simple methods, such as JAC. The results of NN and NN-SINGLE are almost identical; therefore, there is not much effective correlation across the rounds. When the number of class label is 100 (Purchase100, CIFAR100), the success rate of the attack is reduced. In particular, the accuracy of CIFAR100 is low in this case. However, as shown in later, this is surprisingly improved by using a smaller sparse rate.

Figure 4.7 depicts the relationship between the sparse ratio and attack performance. The number of client labels is fixed to two. The results indicate that the sparse ratio is inversely related to the success rate of the attack. This is because the indices of label-correlated gradients become more distinguishable as the sparsity increases. In particular, the case of CIFAR100 demonstrates that the attack

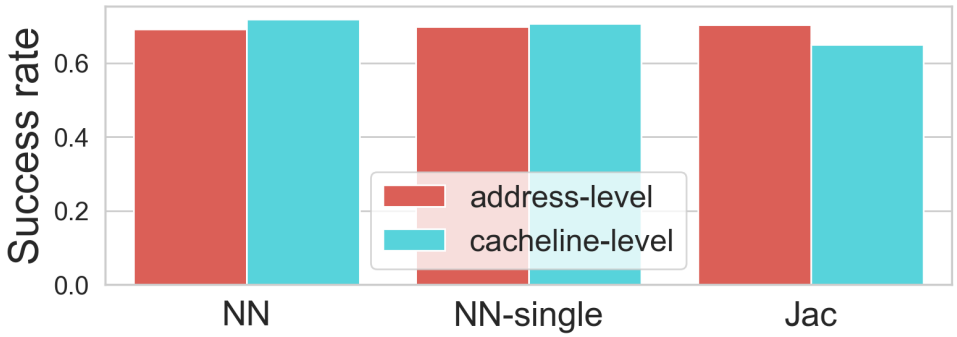


Figure 4.8: Cacheline-level leakage on CNN of CIFAR10: Attacks are possible with at least slightly less accuracy.

is successful only when the sparsity ratio is low. For instance, when the sparsity ratio is 0.3%, the success rate is almost 1.0. Thus, sparsity ratio is an important factor in an attack.

Figure 4.8 depicts a comparison of attack performance based only on index information observed at the cacheline granularity (64 B), which can be easily observed against SGX [84] with CIFAR10 and CNN. The accuracies are almost identical. The NN-based method exhibits slightly higher accuracy, whereas Jac exhibits slightly poorer accuracy. Therefore, the attack is still possible despite observations at the granularity of the cacheline, which indicates that the well-known vulnerability of SGX is sufficient to complete an attack.

Figure 4.9 depicts the evaluation of the size of a dataset required by an attacker to succeed in an attack. The default test dataset accessible to the attacker is presented in Table 4.1—we randomly reduce it on this basis while maintaining the same number of samples for each label. We evaluate the number of labels in the fixed and random labels using the MNIST and Purchase100 datasets, respectively. In MNIST, performance can be preserved even when the amount of data is reduced, which weakens the assumption on dataset size. For example, it is surprisingly noted that, even with 100 samples (i.e., 10 samples per label and 1% of the original evaluation), performance is not affected significantly. On Purchase100, the impact is small, but a meaningful attack is possible with some reduction in data size.

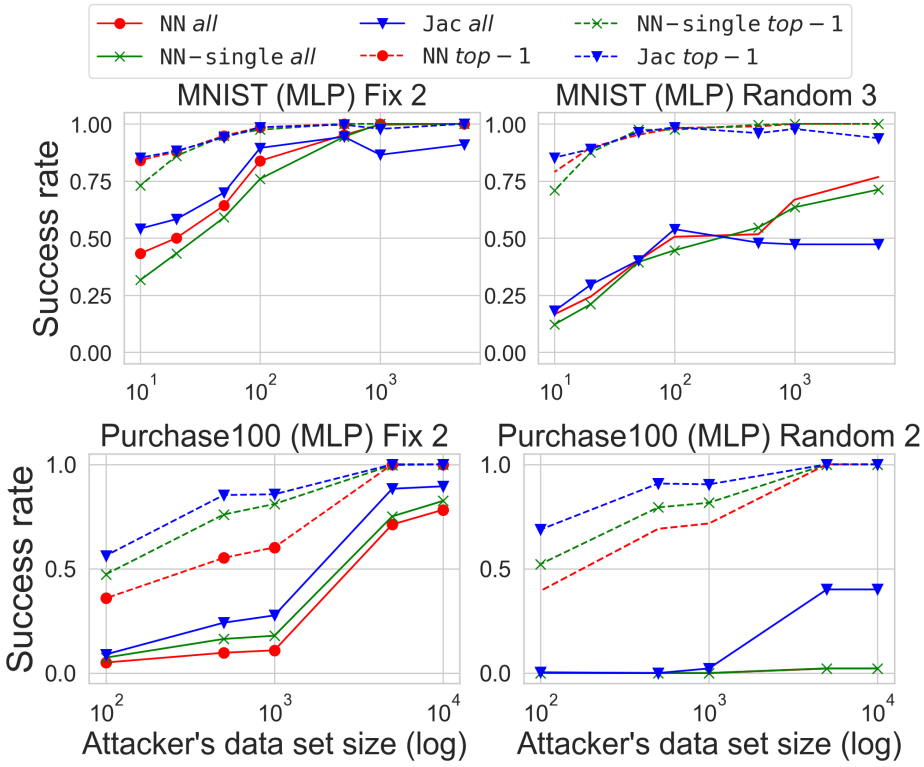


Figure 4.9: The size of data that an attacker needs to access to achieve high success rate can be very small.

4.5 Oblivious Algorithms

In this section, we focus on an aggregation algorithm that can cause privacy leakage, as described in the previous section, and discuss potential avenues of attack prevention. The notation used here is identical to that in Section 4.3.3.

First, we introduce the general ORAM-based method. We initialize ORAM with d zero values for the aggregated parameters, g^*; update the values with the received nk gradients, g, sequentially; and finally retrieve the d values from the ORAM. Because ORAM completely hides memory access to g^*, the algorithm is fully oblivious. However, as established in the experimental section, even the state-of-the-art PathORAM adapted to TEE [163] incurs a significant overhead—thus, a task-specific algorithm is preferable.

4.5.1 Baseline method

Full obliviousness can be simply achieved by accessing all memory addresses to hide access to a specific address. When accessing $\mathbf{G}^*[i]$, a dummy access is performed on $\mathbf{G}^*[j]$ for each $j \in [d]$. For each access, either a dummy or an updated true value is written, and the timing of writing the true value is hidden by an oblivious move `o_mov` (Listing 4.1). The Baseline algorithm is described in Algorithm 12. It accepts the concatenated gradients transmitted by all participants, g (nk-dimensional vector), as input and returns the aggregated gradients, g^* (d-dimensional vector) as output. We make linear accesses to $\mathbf{G}^*$ for a number of times equal to the length of $\mathbf{G}$. Assuming that the memory address is observable at the granularity of the cacheline, as in a traditional attack against the SGX [84], some optimization may be performed. When the weight is four bytes (32-bit floating point) and cacheline is 64 bytes, a 16× acceleration can be achieved. Irrespective of this optimization, the computational and spatial complexities are $O(nkd)$ and $O(nk + d)$, respectively.

Proposition 4. *Algorithm 12 is (cacheline-level) fully oblivious.*

Proof. Let the access pattern observed through algorithm 12 be $\mathbf{Accesses}^{\mathbf{baseline}}$, and it is as follows:

$$\begin{aligned}&\mathbf{Accesses}^{\mathbf{baseline}} = \\ &\quad [(\mathbf{G}[1], \texttt{read}, *), (\mathbf{G}^*_{\mathbf{c}}[1], \texttt{write}, *), ..., (\mathbf{G}^*_{\mathbf{c}}[d/c], \texttt{write}, *), ..., \\ &\quad (\mathbf{G}[k], \texttt{read}, *), (\mathbf{G}^*_{\mathbf{c}}[1], \texttt{write}, *), ..., (\mathbf{G}^*_{\mathbf{c}}[d/c], \texttt{write}, *)]\end{aligned}$$

where c is the number of gradients included in one cacheline and $\mathbf{G}^*_{\mathbf{c}}$ is an array with d/c cells where $\mathbf{G}^*$ is divided at the granularity of a cacheline. Since $\mathbf{Accesses}^{\mathbf{baseline}}$ is the identical sequence for any inputs of the same length, algorithm 12 is 0-statistical oblivious. □

4.5.2 Advanced method

Here, we present a more advanced approach to FL aggregation. In cases with large numbers of model parameters, k and d are significant factors and the computational complexity of the Baseline method becomes extremely high because of the product of k and d. As described in Algorithm 13, we design a more efficient

Algorithm 12 Baseline

Input: $g = g_1 \parallel ... \parallel g_n$: concatenated gradients, nk length
Output: g^*: aggregated parameters, d length

1: initialize aggregated gradients g^*
2: **for** each $(idx, val) \in g$ **do**
3: /* c is the number of weights included in one cacheline */
4: /* offset indicates the position of idx in the cacheline */
5: **for** each $(idx^*, val^*) \in g^*$ if $idx^* \equiv$ offset (mod c) **do**
6: $flag \leftarrow idx^* == idx$ ▷ target index or not
7: $val' \leftarrow$ `o_mov`$(flag, val^*, val^* + val)$ ▷ see `o_mov` (Listing 4.1)
8: write val' into idx^* of g^*
9: **end for**
10: **end for**
11: **return** g^*

Advanced algorithm by carefully analyzing the operations on the gradients. Intuitively, the method is designed to compute g^* directly from the operations on the gradient data, g, to eliminate access to each memory address of the aggregated gradients, g^*. This avoids the overhead incurred by dummy access to g^*, as in the Baseline. The method is divided into four main steps: *initialization* on gradients vector g (Line 1), oblivious sort (Line 4), *oblivious folding* (Line 6), and a second oblivious sort (Line 16). For oblivious sort, we use Batcher's Bitonic Sort [167], which is implemented in a register-level oblivious manner using oblivious swap `o_swap` (Listing 4.2) to compare and swap at all comparators in the sorting network obliviously. Figure 4.10 illustrates a running example for better understanding, where we show a simple example of Algorithm 13 at $n = 3$, $k = 2$ and $d = 4$.

As given by Algorithm 13, we first apply an initialization to g, where we prepare zero-valued gradients for each index between 1 and d (declared g') and concatenate them with g (Lines 1–3). Thus, g has length $nk+d$. This process guarantees that g has at least one weight indexed for each value between 1 and d; however, aggregation of the concatenated g yields exactly the same result as the original g because the added values are all zero. We then apply an oblivious sort to g using the parameter's index (Lines 4–5). Rather than eliminating the connection between the client and gradient, this serves as a preparation for subsequent operations to compute the per-index aggregate values. Next, the *oblivious folding*

Algorithm 13 Advanced

Input: $g = g_1 \parallel ... \parallel g_n$: concatenated gradients, nk length

Output: g^*: aggregated parameters, d length

1: /* *initialization*: prepare zero-valued gradients for each index */
2: $g' \leftarrow \{(1, 0), ..., (d, 0)\}$ ▷ all `values` are zero
3: $g \leftarrow g \parallel g'$ ▷ concatenation
4: /* *oblivious sort* in $O((nk + d) \log^2 (nk + d))$ */
5: oblivious sort g by `index`
6: /* *oblivious folding* in $O(nk + d)$ */
7: $idx \leftarrow$ `index` of the first weight of g
8: $val \leftarrow$ `value` of the first weight of g
9: **for** each $(idx', val') \in g$ **do** ▷ Note: start from the second weight of g
10: $flag \leftarrow idx' == idx$
11: /* M_0 is a dummy index and very large integer */
12: $idx_{prior}, val_{prior} \leftarrow$ `o_mov`$(flag, (idx, val), (M_0, 0))$
13: write $(idx_{prior}, val_{prior})$ into idx' - 1 of g
14: $idx, val \leftarrow$ `o_mov`$(flag, (idx', val'), (idx, val + val'))$
15: **end for**
16: /* *oblivious sort* in $O((nk + d) \log^2 (nk + d))$ */
17: oblivious sort g by `index` again
18: **return** take the first d `values` as g^*

routine is executed (Lines 6–14). It linearly accesses the values of g and cumulatively writes the sum of the values for each index in g. Starting from the first place, it adds each value to the subsequent value if the neighboring indices are identical, and writes a zero-valued dummy index, M_0, in place of the original one. M_0 is a large integer. Otherwise, if the neighboring indices are different, we stop adding values, and the summation of the new index is initiated anew. Thus, we finally obtain g such that only the last weight of each index bears the correct index and aggregated value, and all the remaining ones bear dummy indices. In addition, the initialization process described above guarantees that d distinct indices always exist. In this phase, the index change-points on g during folding are carefully hidden. If the index change-points are exposed, the number corresponding to each index (i.e., the histogram of the indices) is leaked, which can cause catastrophic results. Therefore, oblivious folding employs `o_mov` (Listing 4.1) to make conditional updates oblivious and hide not only the memory

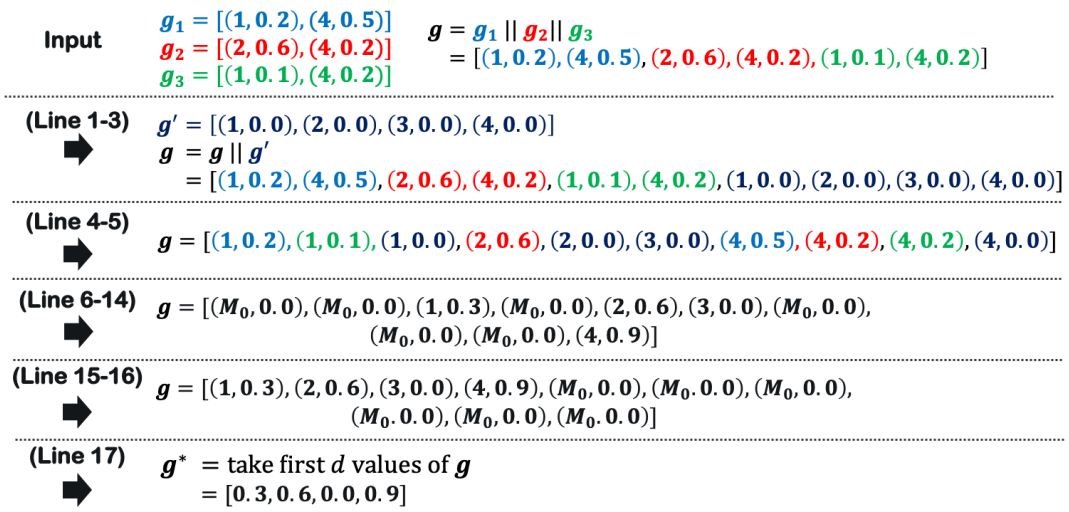


Figure 4.10: Running example of Advanced (Algorithm 13) at $n = 3$ (#user), $k = 2$ (#sparsified dimension), $d = 4$ (#dimension).

access of the data, but also low-level instructions. Finally, we apply an oblivious sort to g (Lines 15–16). After sorting, in g, weights with indices between 1 and d are arranged individually, followed by weights with dummy indices. Finally, taking the `values` of the first d weights of the sorted g, we return this as the final aggregated gradient, g^* (Line 17).

Proposition 5. *Algorithm 13 is fully oblivious.*

Proof. The access pattern, $\mathbf{Accesses}^{\mathbf{advanced}}$, is somewhat complicated, but obliviousness can be considered using a modular approach. Our oblivious sort relies on Batcher's Bitonic Sort, in which sorting is completed by comparing and swapping the data in a deterministic order, irrespective of the input data. Therefore, access patterns generated using this method are always identical. In oblivious folding, the gradient is linearly accessed once; thus, the generated access pattern is identical for all input data of equal length. Finally, $\mathbf{Accesses}^{\mathbf{advanced}}$ are identical and independent of inputs of equal length, this implies 0-statistical obliviousness. □

The complexity of the entire operation is $O((nk + d)\log^2(nk + d))$ in time and $O(nk + d)$ in space. The proposed algorithm relies on an oblivious sort, which dominates the asymptotic computational complexity. We use Batcher's Bitonic Sort [167], which has $O(n\log^2 n)$ time complexity. The Advanced is

asymptotically better than the Baseline because of the elimination of the kd term.

4.5.3 Optimization

In this subsection, we describe an optimization method that fits the basic SGX memory characteristics. The current SGX comprises two major levels of memory size optimization. The first factor is the size of the L3 cache (e.g., 8 MB). In SGX, the acceleration is significant because the cache hit reduces not only the memory access time but also the data-decrypting process. The second factor is the EPC size (e.g., 96 MB). As mentioned in Section 2.3, accessing data outside the EPC incurs serious paging overhead. Compared to the proposed methods, the Baseline is computationally expensive; however, most memory accesses are linear. Thus, it is greatly accelerated by the high cache hit rates and the prefetch functionality of the CPU. However, in Advanced, the low locality of memory accesses in Batcher's sort reduces the cache and EPC hit rates.

Therefore, optimization is performed by introducing a function to split users into appropriate groups before executing Advanced to keep the data processed at one time within the EPC size. This procedure involves the following steps: (1) divide into groups of h users each; (2) aggregate values for each group using Advanced; (3) record the aggregated value in the enclave, and carry over the result to the next group; and (4) only average the result when all groups have been completed and then load them from the enclave to the untrusted area. Note that the improvement to Advanced does not change its security characteristics. An external attacker can only see the encrypted data, and any irregularities in the order or content of the grouped data can be detected and aborted by enclave. The key parameter is the number of people, h, in each group. The overall computational complexity increases slightly to $O(n/h((hk + d)\log^2(hk + d)))$. However, this hides the acceleration induced by cache hits and/or the overhead incurred by repeated data loading. Basically, although lowering h improves the benefit of cache hits, lowering it too much results in a large amount of data loading. The optimal value of h is independent of data and can be explored offline. Our results indicate that there exists an optimal h that achieves the highest efficiency in the experiment.

4.5.4 Relaxation of Obliviousness

We investigate further improvements by relaxing the condition of full obliviousness to achieve better efficiency. A relaxed security definition that has recently garnered attention is that of *differentially oblivious* (DO) [169, 176, 177, 178, 179]. Following the definition of Section 4.2.2, (ϵ, δ)-DO means that for any neighboring inputs I, I', it holds that

$$\Pr[\mathbf{Accesses}^M(\lambda, I)] \leq e^{\epsilon} \Pr[\mathbf{Accesses}^M(\lambda, I')] + \delta.$$

DO is DP applied to obliviousness. This relaxation can theoretically improve the efficiency from full obliviousness. In practice, improvements have been reported for RDB queries [179] whose security model, in which access pattern leakage within the enclave is out of the scope, differs from ours.

However, DO is unlikely to work in the FL setting. DO approaches commonly guarantee DP for the histogram of observed memory accesses. We construct a DO algorithm based on [176, 177]. The procedure involves the following steps: pad dummy data, perform an obvious shuffle (or sorting), and update g^* by performing linear access on $\mathbf{G}$. The observed memory access pattern is equivalent to a histogram of the indices corresponding to all gradients, and the dummy data are required to be padded with sufficient random noise to make this histogram DP. However, this inevitably incurs prohibitive costs in the FL setting. [d] The first reason for this is that the randomization mechanism can only be implemented by padding dummy data [180], which implies that only positive noise can be added, and the algorithms covered by padding are limited (e.g., the shifted Laplace mechanism). The second reason is critical in our case and differs from previous studies [176, 177]. Considering that the ML model dimension, d, and even the sparsified dimension, k, can be large, noise easily becomes significant. For example, considering the DO guaranteed by Laplace noise, where k denotes the sensitivity and d is the dimension of the histogram, the amount of noise is proportional to kd and multiplied by a non-negligible constant, owing to the first reason [176]. This produces huge array data to which oblivious operations must be applied, resulting in a larger overhead than in the fully oblivious case.

[d] We have confirmed this prohibitive cost in our preliminary experiments.

4.5.5 Experimental results

In this section, we demonstrate the efficiency of the designed defense method on a practical scale. Because it is obvious that the proposed algorithms provide complete defense against our attack method, their attack performances are not evaluated here. In addition, our previous algorithms do not degrade utility—the only trade-off for enhanced security is computational efficiency.

Setup: We use an HP Z2 SFF G4 Workstation with an Intel Xeon E-2174G CPU, 64 GB RAM, and 8 MB L3 cache, which supports the SGX instruction set and has 128 MB processor reserved memory, of which 96 MB EPC is available for user use. We use the same datasets as those in Table 4.1 and synthetic data. Note that the proposed method is fully oblivious and its efficiency depends only on the model size. The aggregation methods are the *Non Oblivious* (*linear* algorithm in Section 4.3.3), the *Baseline* (Algorithm 12), the *Advanced*(Algorithm 13), and *PathORAM*. We implement PathORAM based on an open-source library[e] that involves a Rust implementation of Zerotrace [163]. The stash size is fixed to 20. In the experiments, we use *execution time* as an efficiency metric. We measure the time required by an untrusted server from loading the encrypted data to the enclave to completion of aggregation.

Results: Figure 4.11 depicts the execution time for the aggregation operation on the synthetic dataset with respect to model size. α is fixed to 0.01, and the x-axis represents the original model parameter size, d. The proposed *Advanced* is approximately one order of magnitude faster than *Baseline*. Moreover, it is more robust with respect to an increase in the number of parameters. Only when the number of parameters is very small is *Baseline* faster than *Advanced*, because when the model is extremely small, *Baseline*'s simplicity becomes dominant. *PathORAM* also incurs a large overhead. The theoretical asymptotic complexity of the original PathORAM-based algorithm is $O((nk)\log(d))$ because a single update on ORAM can be performed in $O(\log(d))$. However, this is an ideal case and the overhead of the constant factor is large when PathORAM is adapted to the SGX security model (i.e., ZeroTrace [163]). The overhead is primarily induced by the *refresh* operation corresponding to each update and the oblivious reading of the position maps. The result suggests that *PathORAM*'s superiority does not appear until the data size increases hugely. Overall, the

[e] `https://github.com/mobilecoinofficial/mc-oblivious`

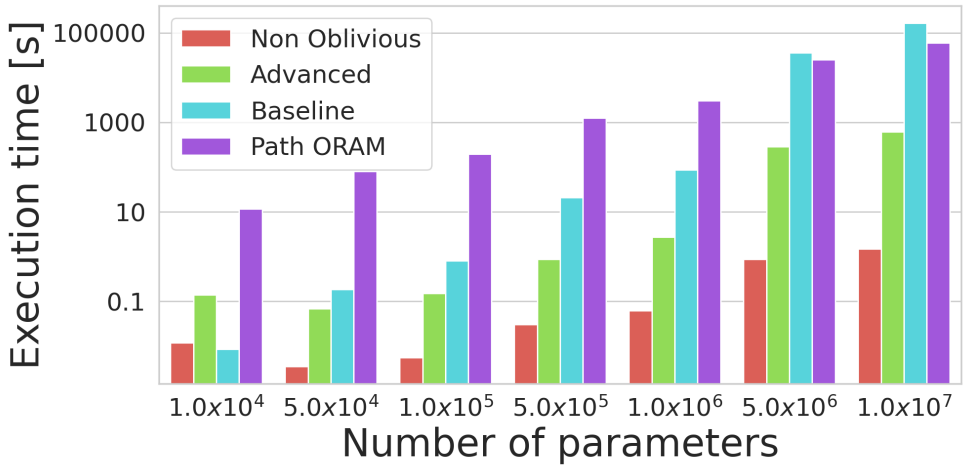


Figure 4.11: Performance results on a synthetic dataset w.r.t. models of various sizes: *Advanced* functions efficiently. α (sparse ratio) $= 0.01$ and n (number of clients per round) $= 100$.

results indicate that the aggregation process can be completed in a few seconds, even if the model scale involves approximately 1M parameters.

Figure 4.12 depicts the performances on MNIST (MLP) corresponding to various numbers of clients and low sparsity ($\alpha = 0.1$). The *Baseline* method is more efficient when the number of clients, N, is large (10^4). Firstly, the model size d is fairly small (i.e., MNIST (MLP) consists of only 50K parameters). Hence, the overhead of the dummy access operations of *Baseline* is not significant. The second reason is that the lower sparsity and higher number of clients increases nk, which increases the overhead for both *Baseline* and *Advanced*, but affects *Advanced* more, as explained by the analysis of cache hits in Section 4.5.3. At $N = 10^4$, the memory size required by *Advanced* is given by (vector to be obliviously sorted) $= 5089 * 8 * 3000 + 50890 * 8 \approx 122$ MB (¿ 96 MB of EPC size) since each cell of gradient is 8 bytes (32-bit unsigned integer for index and 32-bit floating point for value). Batcher's sort requires repeated accesses between two very distant points on the vector, which could require a large number of pagings until *Advanced* finishes; however, in *Baseline*, this hardly occurs. However, the optimization introduced in Section 4.5.3 successfully addresses this problem.

Figure 4.13 illustrates the effects of the optimization method on *Advanced*. The left figure shows the results under the same conditions as the rightmost bars in Figure 4.12 ($N = 10^4$), indicating that *Advanced* is dramatically faster with an

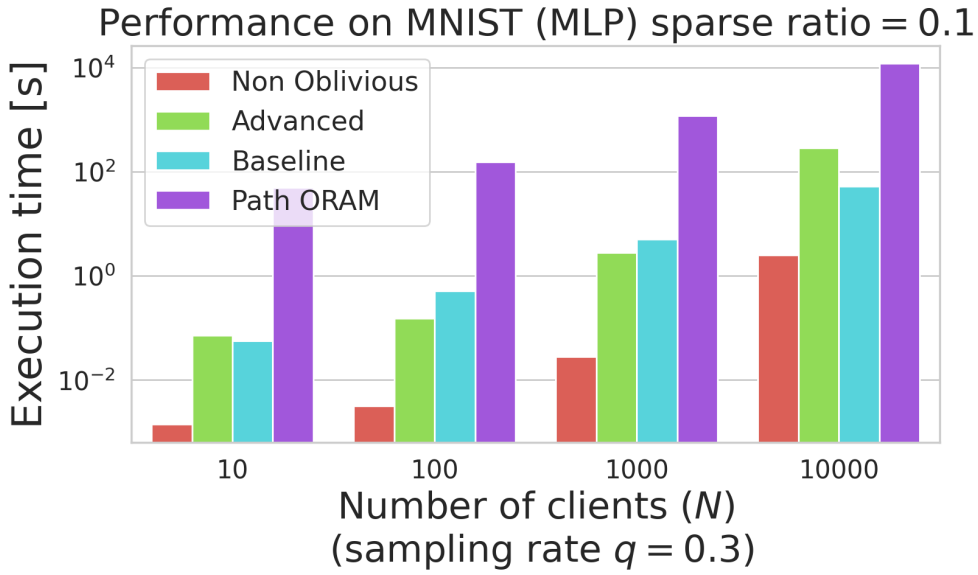


Figure 4.12: Performance results w.r.t. various numbers of clients (N) at low sparsity ($\alpha = 0.1$): the *Advanced* gradually worsens with increasing number of clients.

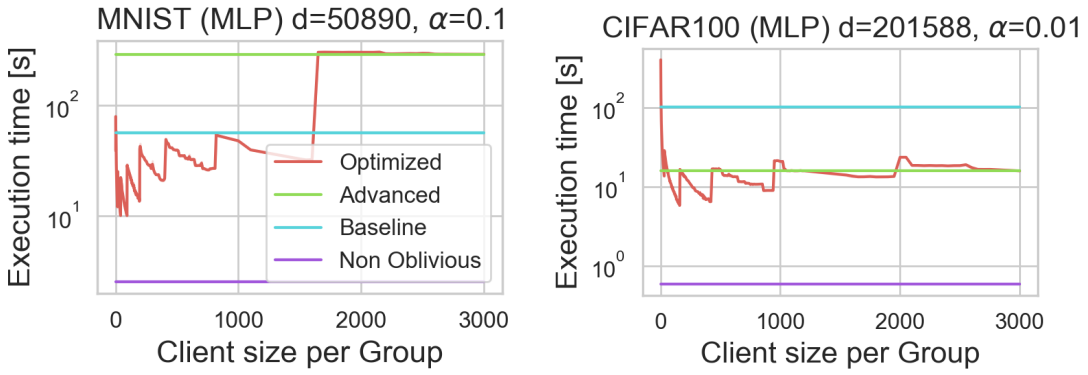


Figure 4.13: The effects of optimizing the *Advanced* on MLP models on MNIST (left) and CIFAR100 (right).

optimal client size. When the number of clients per group, h (represented along the x-axis), is small, the costs of iterative loading to the enclave become dominant, and the overhead conversely increases. However, if h is gradually increased, the execution time decreases. Considering that the size of the L3 cache is 8 MB and data size per user is $d\alpha = 0.04$ MB, the L3 cache can accommodate up to approximately 200 clients. The results of MNIST (MLP) indicate that the lowest

is, approximately 10 s, at around $h = 100$, which is a significant improvement compared to 290 s in the original *Advanced*. The small waviness of the plot appears to be related to the L2 cache (1 MB), which does not have an impact as large as that of the L3 cache. The efficiency decreases significantly around $h = 2000$, owing to the EPC paging. The figure on the right depicts the results on CIFAR100 (MLP) at $\alpha = 0.01$ and $N = 10^4$. In this case, *Advanced* is initially much faster, but there is an optimal h that can be further improved. The pre-optimization execution time of 16 s is reduced to 5.7 s at around 150 clients.

4.5.6 Discussion

Threat assumption. Boenisch et al. [171] reported that *malicious* servers improve inference attack performance beyond *semi-honest*. This type of attack involves crafting global model parameters (called *trap weights* in [171]) and controlling client selection in rounds to highlight the updates of the target user by a malicious server. To prevent parameter tampering, [68] proposed a defense strategy using a cryptographic commitment scheme. The Olive can adopt a similar strategy based on a cryptographic signature. Aggregation is performed within the enclave, and the aggregated global model is signed with the private key in the enclave. This ensures that the model is not tampered with outside the enclave, i.e., by a malicious server. Any client can verify this using a public key which can be easily distributed after RA. In addition, TEE prevents malicious client selection by securely running in the enclave. Therefore, privacy is not violated at least for this type of attack. Other possible malicious server behaviors can influence the security of the Olive, including denial-of-service (DoS) attacks [181], which are outside the threat model of the Olive, as well as TEE and are difficult to prevent.

Security of SGX. Finally, we discuss the use of SGX as a security primitive against known attacks. According to [86], the objectives of attacks against SGX can be classified into the following three: (1) stealing memory/page access patterns or instruction traces [84, 85, 136, 137], (2) reading out memory content [182, 183], and (3) fault injection [184]. (1) is the target of the defense. The speculative execution attacks of (2) are mostly handled by microcode patches. Hence, the protection is usually not required in the application. However, if the microcode is not updated, the gradient information of the enclave may be stolen

by a malicious attacker, which is beyond the scope of this study. The fault injection of (3) is covered within the scope of microcode/hardware [86, 184] and lies outside the security. This may cause DoS even using TEE [181].

In addition, another risk exists if malicious code is embedded in the code executed in the enclave. This can be prevented by verifying the enclave state using RA; however, this requires the source code to be publicly available and assessed. Further, as discussed in [185], the SDK may involve unintended vulnerabilities. To benefit from the security of SGX, the code of TCB must be written properly.

4.6 Relationship with DP-FL.

As discussed in Section 2.2.1, server-side TEE provides utility benefits in DP-FL. For completeness, we explain in detail about the combination of Olive and CDP-FL, because the combination of sparsification and CDP-FL is not such an obvious problem in terms of differential privacy bound.

Algorithm 14 depicts the algorithm for the combination of CDP-FL and Olive. On the client side, after computing the parameter delta, top-k sparsification is executed (Line 21) followed by clipping (Line 22), encryption, and data transmission to the TEE on the server side. This approach just incorporates client-side top-k sparsification into DP-FedAVG [54]. The hyperparameter q is needed for privacy amplification through client-level sampling. σ is the noise multiplier that determines the variance of the Gaussian noise to satisfy DP (Line 12) (which is noise's standard deviation divided by the clipping scale and commonly used in DP-SGD [48] framework). And C is clipping parameter to bound $\ell 2$-sensitivity. A similar procedure has been proposed in [186], although the TEE part is not included.

The privacy analysis of Algorithm 14 is discussed in the rest of this section. Recent works [103, 186] have investigated the combination of client-level CDP-FL and sparsification. The privacy analysis is performed by combining existing Renyi differential privacy (RDP) analysis techniques (or moments accountant [48] which is equivalent to RDP analysis) as well as common CDP-FL [54].

However, one salient aspect is the treatment of sparsification (which is described in Section 4.2.1). The crucial point is whether the indices of the parameters selected by sparsification are common or distinct among all clients. If all clients have common sparsified indices (k out of d indices), the Gaussian mechanism

Algorithm 14 DP-FL in OLIVE

Input: N: # participants, q: sampling rate of participants, η_c, η_s: learning rate, σ: noise parameter, T: number of rounds

KeyStore $\leftarrow$ *Remote Attestation* with all user i ▷ key-value store in enclave that stores sk_i: user i's shared key from RA in provisioning
procedure TRAIN(q, η_c, η_s, σ, T)
    Initialize model θ^0, clipping bound C
    for each round $t = 0, 1, \ldots, T$ **do**
        $\mathcal{Q}^t \leftarrow$ (sample users with probability q) ▷ securely in enclave
        for each user $i \in \mathcal{Q}^t$ **in parallel do**
            $\text{Enc}(\Delta_i^t) \leftarrow \text{ENCCLIENT}(i, \theta^t, \eta_c, C)$ ▷ with AE mode
            LoadToEnclave($\text{Enc}(\Delta_i^t)$)
            check if user i is in $\mathcal{Q}^t$
            $sk_i \leftarrow$ KeyStore[i] ▷ retrieve user i's shared key
            $\Delta_i^t \leftarrow \text{Decrypt}(Enc(\Delta_i^t), sk_i)$ ▷ with verification
        end for
        /* **Obliviously performed, such as Alg. 12 or 13** */
        $\tilde{\mathbf{\Delta}}^{\mathbf{t}} = \frac{\mathbf{1}}{\mathbf{qN}} \left(\sum_{\mathbf{i} \in \mathcal{Q}^{\mathbf{t}}} \mathbf{\Delta_i^t} + \mathcal{N}(\mathbf{0}, \sigma^{\mathbf{2}} \mathbf{C^2} \mathbf{I_d}) \right)$ ▷ oblivious aggregation
        LoadFromEnclave($\tilde{\Delta}^t$)
        $\theta^{t+1} \leftarrow \theta^t + \eta_s \bar{\Delta}^t$
    end for
end procedure
procedure ENCCLIENT(i, θ^t, η, C)
    $\theta \leftarrow \theta^t$
    $\mathcal{G} \leftarrow$ (user i's local data split into batches)
    for batch $g \in \mathcal{G}$ **do**
        $\theta \leftarrow \theta - \eta \nabla \ell(\theta; g)$
    end for
    $\Delta \leftarrow \theta - \theta^t$
    $\Delta \leftarrow \text{TopkSparse}(\Delta)$ ▷ top-k sparsification
    $\Delta' \leftarrow \Delta \cdot \min\left(1, \frac{C}{||\Delta||_2}\right)$ ▷ $\ell 2$ clipping
    $\text{Enc}(\Delta') \leftarrow \text{Encrypt}(\Delta', sk_i)$ ▷ with shared key sk_i from RA
    return $\text{Enc}(\Delta')$
end procedure

required for DP only needs k-dimensional noise, as only k parameters of the global model require updating in a single round of aggregation. This results in a direct reduction of noise by a factor of $O(k/d)$. To this end, [103] proposes a method for obtaining the common top-k indices among many clients for sparsification. However, as noted in [97], in a practical setting, there is actually little overlap in the top-k sparsified indices for each client, especially in the non-i.i.d. setting, which is general in FL. Hence, a common top-k index appears to be impractical.

On the other hand, we consider the scenario where different sparsified indices are chosen for different clients. This represents a standard setup in the absence of DP. In contrast to the previous case, where all clients shared a common set of sparsified indices, there is no reduction in Gaussian mechanism noise on the order of $O(k/d)$. This is due to the fact that while each client transmits sparsified parameters of dimension k, any of the d dimensions of the global model may be updated with the transmitted sparsified parameters. Hence, noise needs to be added to all dimensions to ensure DP rather than only to the k dimensions. This remains true regardless of whether the noise is added on the client or server side, or what type of sparsification is employed as far as aiming to guarantee a global model DP. This may have been overlooked in previous work that employed sparsification [122].

Nevertheless, despite the above discussion, such client-specific sparsification can improve the trade-off between privacy and utility to a certain extent. This is because sparsification reduces the absolute value of the $\ell 2$-norm of the transmitted parameters. As we formally describe later, the $\ell 2$-norm of the shared parameters from each client must be bounded by the clipping parameter C to add Gaussian noise for DP. When clipping is performed on the original dense parameters, all parameters contribute to the $\ell 2$-norm. In the case of sparsification, however, only k parameters contribute to the $\ell 2$-norm. Intuitively, the less important $d - k$ parameters are discarded and the space in the $\ell 2$-norm is allocated to the more important k parameters, thus increasing their utility. Consequently, this also means that the clipping size C can be set lower in the sparsified case, which can lead to lower noise variance. This observation is the basis for the sparsification proposed in [186]. To be more precise, [186] sparsifies according to their own utility criteria, rather than selecting the top-k parameters, but the characteristics of the privacy-utility trade-offs are the same. In general, it can be concluded that the amount of noise required for CDP is the same in the case of

sparsification as in the absence of sparsification.

Formal privacy statement. We now formally state the DP satisfied by Algorithm 14 for completeness. The following definitions and lemmas are the same as the ones stated in existing studies such as [103, 186].

We use Rényi DP (RDP) [50] because of the tightness of the privacy analysis and the composition.

Lemma 7 (RDP to DP conversion [49])**.** *If $\mathcal{M}$ satisfies (α, ρ)-RDP, then it also satisfies $(\rho + \frac{\log(1/\delta)}{\alpha-1}, \delta)$-DP for any $0 < \delta < 1$.*

Here, we state the formal differential privacy guarantees provided by Alg. 14.

Theorem 7. *For any $\epsilon < 2\log(1/\delta)$ and $0 < \delta < 1$, Alg. 14 satisfies (ϵ, δ)-DP after T communication rounds if*

$$\sigma^2 \geq \frac{7q^2T(\epsilon + 2\log(1/\delta))}{\epsilon^2}.$$

Proof. In each round t of T in TRAIN (Line 2 of Alg. 14), let f be a summation of delta parameters (Δ_i^t, Line 11), the $\ell 2$-sensitivity of f is C due to clipping operation (Line 22). As explained in detail above, this is independent of the sparsified dimension k. Hence, adding the Gaussian noise $\mathcal{N}(0, \sigma^2C^2\mathbf{I}_d)$, i.e., G_f, satisfies $(\alpha, \alpha/2\sigma^2)$-RDP for any $\alpha > 1$ by Lemma 3. Further, in the round, the participants are sub-sampled with probability q (Line 5). Then, following Lemma 3 of [187], if $\sigma^2 \geq 0.7$ and $\alpha \leq 1 + (2/3)C^2\sigma^2 \log \frac{1}{q\alpha(1+\sigma^2)}$, by Lemma 4, sub-sampled Gaussian mechanism $G'_f(\cdot)$ satisfies $(\alpha, \frac{3.5q^2\alpha}{\sigma^2})$-RDP. Over T rounds, by Lemma 1, it satisfies $(\alpha, T\frac{3.5q^2\alpha}{\sigma^2})$-RDP. Lastly, we convert RDP guarantee to (ϵ, δ)-DP by Lemma 7. ϵ needs to hold $T\frac{3.5q^2\alpha}{\sigma^2} + \frac{\log 1/\delta}{\alpha-1} \leq \epsilon$. Choose $\alpha = 1 + 2\log(1/\delta)$, we obtain the final result. □

4.6.1 Attack evaluation

Here, we demonstrate that the proposed attack remains viable even in the presence of DP. Firstly, we elucidate the reasons for the attack circumventing DP in Algorithm 14. During each round of FL, the attacker is able to observe the index prior to perturbation (Line 12 of Algorithm 14), thereby exposing the raw index information. It should be noted that CDP-FL also employs distributed Gaussian noise on the client side. However, it is performed after sparsification [186], which

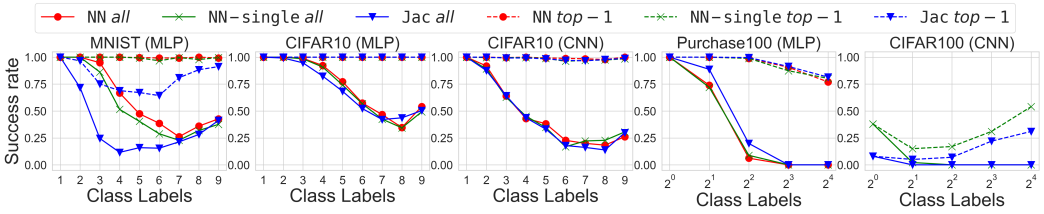


Figure 4.14: Attack results on datasets with a fixed number of labels with DP ($\sigma = 1.12$).

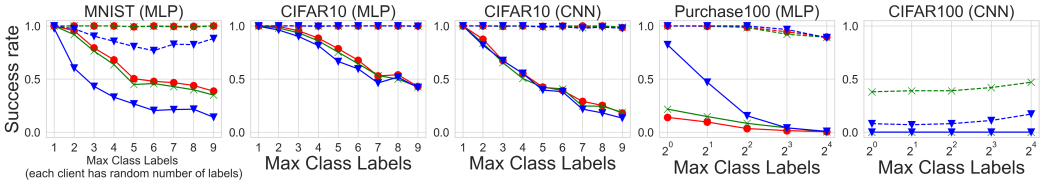


Figure 4.15: Attack results on datasets with a random number of labels (more difficult setting) with DP ($\sigma = 1.12$).

implies that the raw index information is still visible. Nevertheless, the randomization of the parameters of the global model by DP may reduce the accuracy of the attack. This approach should be considered carefully, as the model may not be well trained itself. In the next experiment, we see how much protection and how much model utility is sacrificed by the DP-based approach.

The experimental setting is the same as Section 4.4.3. When the noise multiplier σ is set to 1.12, the attack is essentially unaffected. Figures 4.14 and 4.15 are DP versions of Figures 4.5 and 4.6. Although the success rate of attacks has decreased somewhat, there is almost no change. Attacks are still possible.

In Figure 4.16, we show the attack results on MLP of MNIST for increasing noise scale with fixed number of labels 3. The horizontal axis indicates noise scale σ by DP and the left-side start points indicate no noise. Compared to the case with no noise, increasing the noise has less effect on the attack performance. This makes sense from our attack design, where the attacker observes the raw index information of gradients even though the global model satisfies DP. The blue line in the figure shows the attack success rate for oblivious algorithm (i.e., random inference by the attacker). Since the number of labels is fixed at 3 and the total number of labels is 10, the success rate of this attack is $1/{}_{10}C_3 < 0.01$. We can see that there is a limit to the defensive performance of the DP. When we increase the noise multiplier (σ is over 4.0), defensive performance starts to

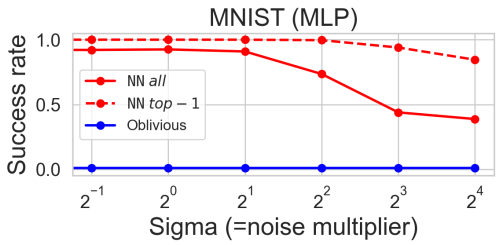


Figure 4.16: Attack performance with variable noise multiplier σ. At realistic noise scales, the attack performance remains high.

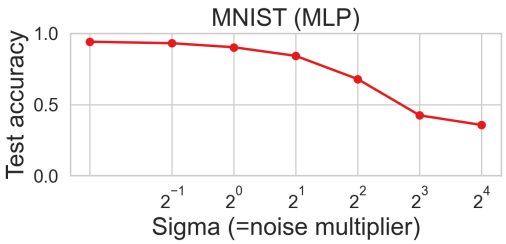


Figure 4.17: Effective noise scales in defensing do not provide sufficient utility.

increase, but such noise multiplier is over-strict in practical privacy degree. This can be seen in Figure 4.17. The figure shows the utility of the models trained with each noise multiplier, plotting the test accuracy when training MNIST with the MLP model. The number of training rounds are fixed at 300, which is based on the observation that the training loss increased and did not converge with large multipliers (Figure 4.18). The results show that models trained with large noise multipliers are no longer useful, and that realistic noise does not protect against attacks. These results highlight the importance of OLIVE even in CDP-FL.

4.7 Related works

Positioning in Security and Privacy threats. FL contains many attack surfaces because of its decentralized and collaborative scheme. These can be broadly classified into inference attacks by semi-honest parties [175, 23, 166] and attacks that degrade or control the quality of the model by malicious parties [188, 71, 170]. However, [171] demonstrated that malicious servers may enable effective inference attacks by crafting aggregated parameters. Our target is taken to be an inference attack by a semi-honest server. Inference attacks include reconstruction [174, 189], membership [175], and label inferences [23, 166]. In particular, it has been reported that shared parameters observed by a server contain large amounts of private information [20, 21]. Our work targets gradient-based label inference attacks, [23, 166] use the gradients themselves, focusing on the values, and not only on the indices leaking from the side-channel, as in our

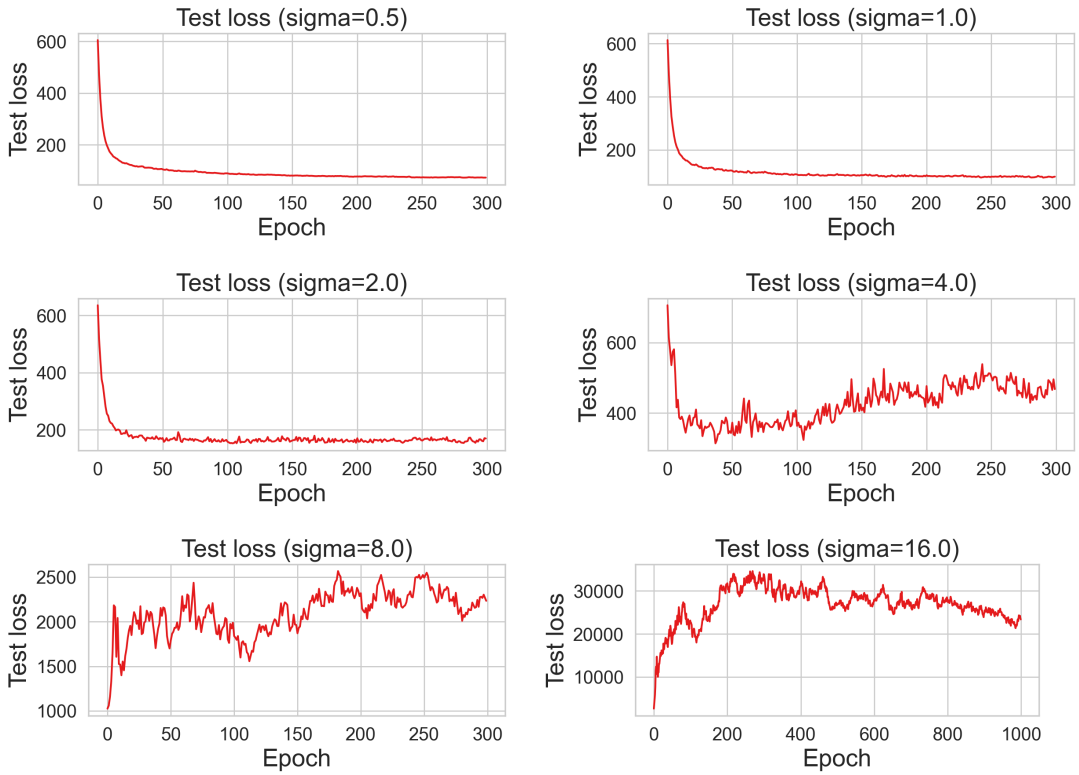


Figure 4.18: Test losses for each noise multiplier σ.

method. To the best of our knowledge, this is the first study to demonstrate label inference using only sparsified index information.

Relationship with SA. Secure aggregation (SA) [67] is a popular FL method for concealing individual parameters from the server and it is based on the lightweight pairwise-masking method [96, 66, 97], homomorphic encryption [98, 99] or TEE [74, 71]. In this study, we study SA using TEE. Recent studies have investigated combinations of SA and sparsification, such as random-k [97] and top-k [161]. However, these are not in harmony because they require the same sparsified indices among clients for mask cancellation. [161] proposed generation of common masks by taking a union set of top-k indexes among clients, which incurs extra communication costs and strong constraints. This can be serious for the top-k because, in fact, Ergun et al. [97] showed that the top-k indices exhibits little overlap between clients, which is especially noticeable in the non-i.i.d. as in FL. In [97], only a pair of users exhibited a common index; however, this was applicable only to random-k sparsification. In the case of TEE, a common index

or random-k is not required; but, individual indices can still be leaked through side-channels. Therefore, our work focuses on attacks and defense strategies at this point.

Oblivious techniques. The oblivious algorithm [87, 162, 140] is known to induce only independent memory access patterns for the input data. Although PathORAM [162] is the most efficient ORAM implementation, it assumes a private memory space of a certain size (called as *client storage*) and is not applicable to Intel SGX [163]. Zerotrace [163] adapted PathORAM to the SGX security model, in which the register is only private memory. The authors used the oblivious primitive proposed in [140], in which the program did not leak instruction sequences from the CPU register, using x86 conditional instructions. Our proposed algorithm also uses the low-level primitives; however, high-level algorithms are considerably different. [164] studied oblivious SQL processing. Their proposal included a *group-by* query, which is similar to our proposed algorithm in concept. The aggregation algorithm computes the summed dense gradients based on multiple sparse gradients, which can be viewed as a special case of the *group-by* query. However, our method is more specialized, for instance, we first prepare the zero-initialized dense gradients to hide the all of index set that are included and then obliviously aggregated, which is impossible in the case of *group-by*. In addition, the aforementioned algorithms are fundamentally different because they focus on the data distributed across nodes. Further, [164] did not consider the technique proposed by [140] for linear access, which can induce additional information leaks in the conditional code [84]. [190, 191] studied compiling and transforming approaches from high-level source code to low-level oblivious code. They proposed a compiler that automatically identifies non-oblivious parts of the original source code and fixes them. However, the authors did not provide customized high-level algorithms for specific purposes, unlike our method. The Differentially Obliviousness (DO) [169, 176, 179] is described in detail in Section 4.5.4.

4.8 Conclusion

In this study, we focused on the integration of server-side TEE into FL as a promising way to address the weakness of privacy and security against untrusted server in FL. We have revealed that, in fact, a gap exists for that purpose: the

vulnerability of TEE. We analyzed the risks of FL with server-side TEE in a sparsified gradient setting, and designed and demonstrated a novel inference attack using gradient index information that is observable from side-channels. To mitigate these risks, we proposed an oblivious federated learning system, called the Olive, by designing fully oblivious but efficient algorithms. The experimental results demonstrated that the proposed algorithm is more efficient than the state-of-the-art general-purpose ORAM and can serve as a practical method on a real-world scale. We believe that our study is useful for realizing privacy-preserving FL using a TEE.

4.8.1 Future works

- This research has only shown that attacks are possible through observation of memory access patterns, and one future direction would be to explore how far the assumptions and goal of the attacker can be extended, such as if the memory access patterns contain a lot of noise, or if more critical privacy attacks are possible beyond the sensitive label leakage.

- For a more efficient oblivious algorithm, we sought a relaxed oblivious definition, DO, but very recent work suggests that fully oblivious sort may be even faster under the assumptions of SGX based on Waksman Network [192] instead of Bitonic sorting Network. In particular, FL would benefit from the asymptotic speedup of oblivious sort by offline computation as proposed in [193] because FL involves a periodic synchronous process.

- This study provides no protection against malicious clients. On the other hand, some of the studies also consider the direction to deal with malicious clients by introducing client-side TEE [73]. We believe that the assumption of client-side TEE is a bit too strong, however, in Cross-silo FL, client-side TEE may be acceptable [194]. Therefore, exploring how TEE can be used to defend against malicious clients is an interesting direction. Also, it would be practical and interesting to investigate the combination with lighter cryptographic techniques would provide protection against malicious client attacks in FL.

4.9 Appendix

4.9.1 Model architectures

Here are some details about the neural network model we used in the experiments. The code for all models is available from our public repository.

Table 4.2 describes the detailed design of the model used in the neural network-based attack in section 4.4.3. Table 4.3 shows the model used as the FL's global model throughout all experiments.

Table 4.2: Architectures of the neural networks used in Section 4.4. d is the number of parameters of the global model trained in FL and $|L|$ is the number of labels of inference target.

Name	Layers	Details
NN	2 Fully Connected Layers	Input: d Hidden: 1000 Dropout: 0.5 Activation: ReLU Output: $\|L\|$
NN-Single	2 Fully Connected Layers	Input: d Hidden: 2000 Dropout: 0.5 Activation: ReLU Output: $\|L\|$

Table 4.3: Architectures of the neural networks used as global models in all FL experiments in Sections 4.4.3 and 4.5.5. Readers can find the details of *ResNet-18* at `https://github.com/weiaicunzai/pytorch-cifar100/blob/master/models/resnet.py`.

Name	Layers	Details
MNIST MLP	2 Fully Connected Layers	Input: 28 * 28 Hidden: 64 Dropout: 0.5 Activation: ReLU Output: 10
CIFAR10 MLP	2 Fully Connected Layers	Input: 3 * 32 * 32 Hidden: 64 Dropout: 0.5 Activation: ReLU Output: 10
CIFAR10 CNN	Convolutional 1	Input: 3 * 32 * 32 Activation: ReLU Maxpooling: kernel size: 2 stride: 2
	Convolutional 2	Input: 6 * 14 * 14 Activation: ReLU Maxpooling: kernel size: 2 stride: 2
	3 Fully Connected Layers	Input: 16 * 5 * 5 Hidden1: 120 Activation: ReLU Hidden2: 84 Activation: ReLU Output: 10
Purchase100 MLP	2 Fully Connected Layers	Input: 600 Hidden: 64 Dropout: 0.5 Activation: ReLU Output: 100
CIFAR100 CNN	*ResNet-18*	

CHAPTER 5

VLDP: Preventing Manipulation Attack in Local Differential Privacy Using Verifiable Randomization Mechanism

The third framework presented in this chapter concerns malicious clients i.e., **E**, which none of the previous frameworks address, as shown in Table 1.2. Research on malicious clients in distributed computing environments is generally very difficult because we need to make sure for the integrity against untrusted remote devices. Therefore, we focus on simpler FA rather than FL to pioneer research on client-side security techniques. Importantly, FA with LDP and FL have the same structure in terms of attacks by malicious clients. Both have operations that pre-processes raw data on the client side (i.e., LDP mechanism in FA with LDP, and local training in FL) and aggregates the pre-processed data on the server side. As described in detail in this chapter, under this structure, the malicious client has two possible attacks: one is to replace the input data itself with maliciously crafted data, and the other is to directly fake the output data. We

have distinguished these as *input-manipulation* and *output-manipulation* in the context of FA with LDP, but it should be emphasized that a similar classification exists in FL. In the context of FL, these are also known as *model update poisoning* and *data poisoning attacks* [93]. Therefore, we believe the consideration of attack and defense in this study could be structurally applicable to FL. Furthermore, it should be emphasized that since it is not possible to assume a powerful TEE with RA capabilities such as Intel SGX on the client side of the edge, we focus on alternative MPC technologies to provide client-side verifiability, which can be implemented in a software-based approach. Finally, our proposed defense in this chapter is not perfect for the input-manipulation, thus, the marker in Table 1.2 is Partial (◗).

5.1 Introduction

Local differential privacy (LDP) [42] (originally [195]) is a promising privacy-enhanced technique for collecting sensitive information without a trusted data curator. Each client perturbs sensitive data locally by a randomized mechanism satisfying differential privacy. A server can run analysis such as frequency estimation based on the perturbed data without accessing the raw data. We can see the effectiveness and feasibility of LDP in early production releases of the platformers such as Google [196], Apple [197], and Microsoft [198], which all utilize LDP for privacy-preserved data curation.

While many studies have been focusing on improving the utility of LDP protocols[a] [199, 200, 201, 202, 203, 204] in the literature, recent studies [39] and [38] report a vulnerability of LDP protocol and alert the lack of security. Specifically, [39] and [38] show that malicious clients can manipulate the analysis, such as frequency estimation, by sending false data to the server. Malicious clients can skew the estimations effectively by considering that estimations are calculated by normalizing with randomization probability defined in the LDP protocol and can even control the estimations. Their studies significantly highlight the necessity of a secure LDP protocol to defend against malicious clients. The problematic point

[a]In this chapter, the defined interactions between clients and servers that take place in Federated Analytics [19] that satisfy LDP are referred to as LDP protocols, such as frequency estimation with guaranteed LDP. In a distributed setting, the server does not collect raw data from clients, but rather collects preprocessed information and estimates statistics.

of protecting against such an attack is that, in a general LDP protocol, others cannot verify the integrity of results without the original data. The randomization would provide data providers plausible deniability for their outputs.

To the best of our knowledge, no effective way in the literature can *completely* prevent manipulation attacks. Although Cao et al. [39] showed some of the countermeasures against malicious clients, their empirical results showed that preventing against *output manipulation attack* is still an open problem. Among their proposed methods, the one normalizing the estimated probability distribution was shown to be to some extent effective for input-manipulation (i.e., the attackers can falsely manipulate the raw input data but honestly execute the local randomized mechanism). However, the proposed countermeasures in [39] are not very effective for output-manipulation attack (i.e., the attackers can arbitrarily change the output of the local randomized mechanism). In addition, their detection-based countermeasures are based on the assumption of specific attack methods and may not be effective against arbitrary output-manipulation attacks. The authors concluded the need for more robust defenses against these attacks. Concurrently, Cheu et al. [38] also emphasize the same conclusion for manipulation attacks they call. There is another promising direction against an attacker who exploits the random mechanism of Differential privacy. Narayan et al. [205] propose an interesting scheme to prove integrity for executing correct randomization mechanisms for Differential privacy. However, their setting is different from ours since they focus on central DP with the data curator, who has the sensitive data, and the analyst, who creates the proof (in this setting, the client needs to prove their local execution).

To solve these problems, we design a novel verifiable LDP protocol based on Multi-Party Computation (MPC) techniques in this work. Our contributions are summarized below. First, we categorize the attacks of malicious clients into two classes, *output-manipulation* and *input-manipulation* (formally defined in Section 5.3). For input-manipulation attacks, efficient countermeasures have been provided in [39], but existing studies cannot prevent output-manipulation wholly and effectively. We analyze the effectiveness of output-manipulation compared to input-manipulation, highlight the importance of output-manipulation protection, and formalize the definition of *output-manipulation secure* LDP protocol. Second, we propose secure and efficient verifiable LDP protocols to prevent manipulation attack. The proposed protocols enable the server to verify the com-

pleteness of executing an agreed randomization mechanism on the client side without sacrificing local privacy. Specifically, we leverage Cryptographic Randomized Response Technique (CRRT)[206] as a building block to convert existing state-of-the-art LDP mechanisms including kRR [199], OUE [201], and OLH into output-manipulation secure LDP protocols with negligible utility loss. Our proposed secure protocols do not assume any specific attack, and work effectively against general output-manipulation, and thus are more potent than previously proposed countermeasures. Third, we conduct intensive experiments to test the performance of the proposed protocols. We demonstrate that the proposed methods can completely protect the LDP protocol from output manipulation attacks with acceptable computational overhead.

The main contribution is as follows:

- **Attack classification**: We classify attacks of malicious clients in LDP into two classes and suggest that different types of countermeasures are needed.
- **Verifiable LDP protocol**: We propose a novel verifiable LDP protocol for three state-of-the-art randomization mechanisms, kRR, OUE, OLH. Our proposed method completely prevents malicious clients' *output-manipulation*.
- **Implementation and evaluation**: We implement the proposed protocol, open the source code, and evaluate our method using the prototype.

In particular, our proposed method works for defending *output-manipulation* attacks. We leave the countermeasures for *input-manipulation* as future work.

5.2 Attacks on LDP protocols

5.2.1 Local Differential Privacy

Differential privacy (DP) [42] is a rigorous mathematical privacy definition, which quantitatively evaluates the degree of privacy protection when we publish outputs about sensitive data in a database. DP is a central model where a trusted server collects sensitive data and releases differentially private statistical information to an untrusted third party. On the other hand, Local DP (LDP) is a local model, considering an untrusted server that collects clients' sensitive data. Clients

perturb their data on their local environment and send only randomized data to the server to protect privacy.

In this work, we suppose server S collects data and aggregates them, and N clients c_i $(0 \leq i \leq N-1)$ send their sensitive data in a local differentially private manner. Each client has an item v which is categorical data, and the items have d domains and $v \in [0, d-1](:= [d])$. Additionally, v_i denotes c_i's item. The clients randomize their data by randomization mechanism $\mathcal{A}$, and c_i sends $\mathcal{A}(v_i) = y_i (\in D)$ to the server, where D is the output space of $\mathcal{A}$. The server estimates some statistics by $\mathcal{F}(y_0, ..., y_{N-1})$. In particular of this work, $\mathcal{F}_k$ corresponds to *frequency estimation* for item k (i.e., how many clients have chosen item k). The formal LDP definition is as follows:

Definition 9 (ϵ-local differential privacy (ϵ-LDP))**.** *A randomization mechanism $\mathcal{A}$ satisfies ϵ-LDP, if and only if for any pair of input values $v, v' \in [d]$ and for all randomized output $y \in D$, it holds that*

$$\Pr[\mathcal{A}(v) \in y] \leq e^{\epsilon} \Pr[\mathcal{A}(v') \in y].$$

Under a specific randomized algorithm $\mathcal{A}$, we want to estimate the frequency of any items. Wang et al. [201] introduce "pure" LDP protocols with nice symmetric property and a generic aggregation procedure to calculate the unbiased frequency estimations from given randomization probabilities. Let $\mathsf{Support}$ be a function that maps each possible output y to a set of input that y supports. $\mathsf{Support}$ is defined for each LDP protocol, and it specifies how the estimation can be computed under the LDP protocol. A formal definition of pure LDP is as follows:

Definition 10 (Pure LDP [201])**.** *A protocol given by $\mathcal{A}$ and $\mathsf{Support}$ is pure if and only if there exist two probability values $p > q$ such that for all v_1,*

$$\begin{aligned} \Pr[\mathcal{A}(v_1) \in \{o | v_1 \in \mathsf{Support}(o)\}] &= p, \\ \forall_{v_2 \neq v_1} \Pr[\mathcal{A}(v_2) \in \{o | v_1 \in \mathsf{Support}(o)\}] &= q. \end{aligned} \tag{5.1}$$

where p, q are probabilities, and q must be the same for all pairs of v_1 and v_2.

While maximizing p and minimizing q make the LDP protocol more accurate, under ϵ-LDP it must be $\frac{p}{q} \leq e^{\epsilon}$. The important thing is that, in pure LDP protocol, we can simply estimate the frequency of item k as follows:

$$\mathcal{F}_k = \frac{\sum_i \mathbb{1}_{\mathsf{Support}(y^i)}(k) - Nq}{p - q} \tag{5.2}$$

We can interpret that this formula normalizes observed frequencies using probabilities p and q to adjust for randomization.

For frequency estimation under LDP, we introduce three state-of-the-art randomization mechanisms, kRR [199], OUE [201] and OLH [201]. These mechanisms includes three steps: (1) Encode is encoding function: $\mathcal{E} : v(\in [d]) \to v'(\in [g])$, (2) Perturbation is randomized function: $\mathcal{A} : v'(\in [g]) \to y(\in D)$, (3) Aggregation calculates estimations from all collected values: $\mathcal{F} : (y_0, ..., y_{N-1}) \to \mathbb{R}$. Formal proofs that each protocol satisfies ϵ-LDP can be found in [201].

k-ary Randomized Response (kRR) is an extension of Randomized Response [207] to meet ϵ-LDP. In particular, kRR provides accurate results in small item domains. This mechanism does not require any special encoding, and provides an identity mapping $\mathcal{E}(v) = v$ ($[g] = [d]$). Perturbation is as follows;

$$Pr[\mathcal{A}(v) = y] = \begin{cases} p = \dfrac{e^\epsilon}{e^\epsilon + d - 1}, & \text{if } y = v \\ q = \dfrac{1-p}{d-1} = \dfrac{1}{e^\epsilon + d - 1}, & \text{if } y \neq v \end{cases} \tag{5.3}$$

For aggregation, we can consider Support function as $\mathsf{Support}(v) = (v)$ and make this follow pure LDP protocol (Definition 10). Therefore, aggregation follows Eq.(5.2).

Optimized Unary Encoding (OUE) encodes item v into d-length bit vector and encode function is defined as $\mathcal{E}(v) = [0, ..., 0, 1, 0, ..., 0]$ where only single bit corresponding to v-th position is 1. Final output space is also d dimensional bit vector $\{0, 1\}^d$ (e.g. $\mathbf{y} = [1, 0, 1, 1, 0]$). Let i-th bit of output vector as y_i, perturbation is as follows;

$$Pr[y_i = 1] = \begin{cases} p = \dfrac{1}{2}, & \text{if } i = v \\ q = \dfrac{1}{e^\epsilon + 1}, & \text{if } i \neq v \end{cases} \tag{5.4}$$

These p and q minimize the variance of the estimated frequency in similar bit vector encoding (e.g. RAPPOR [196]). In aggregation step, we consider Support function as $Support(\mathbf{y}) = \{v | y_v = 1\}$, and also calculate using Eq.(5.2).

Optimized Local Hashing (OLH) employs hash function for dimensional reduction to reduce communication costs. It picks up H from a universal hash

function family $\mathbb{H}$, and H maps $v \in [d]$ to $v' \in [g]$ where $2 \leq g < d$. Therefore, encode function is $\mathcal{E}(v) = H(v)$. Perturbation is the same as kRR, except that the input/output space is $[g]$. Then, p and q is defined as follows;

$$Pr[\mathcal{A}(x) = y] = \begin{cases} p = \dfrac{e^\epsilon}{e^\epsilon + g - 1}, & \text{if } y = H(v) \\ q = \dfrac{1}{g} \cdot p + \left(1 - \dfrac{1}{g}\right) \cdot \dfrac{1}{e^\epsilon + g - 1} = \dfrac{1}{g}, & \text{if } y \neq H(v) \end{cases} \tag{5.5}$$

In aggregation step, we consider Support function as $Support(\mathbf{y}) = \{v | v \in [d] \text{ and } y = H(v)\}$ and follow Eq.(5.2) using p and q.

5.2.2 Attacks on LDP protocols

In this subsection, we introduce two important studies suggesting caution to the necessity of secure LDP protocols.

Targeted Attack. Cao et al. [39] focus on *targeted* attacks to LDP protocols, where the attacker tries to promote the estimated frequencies of a specific item set. Considering the attacker against the LDP protocols, M malicious clients, who can arbitrarily control local environments and send crafted data to the server, are injected by the attacker. (They call *data poisoning attacks*.) The attacker wants to promote r target items $T = \{t_1, ..., t_r\}$ in the frequency estimation. Cao et al. propose three attacks: Random perturbed-value attack (RPA), Random item attack (RIA), Maximal gain attack (MGA). The first two attacks are as baselines and MGA is an optimized attack. In RIA, malicious clients perform uniform random samplings of a value from the target item set. And then, following the LDP protocol, encoding and perturbation are performed and sent to the server. MGA is more complicated than others. It aims to maximize the attacker's overall gain G: sum of the expected frequency gains for the target items, $G = \sum_{t \in T} \mathbb{E}[\Delta f_t]$ where Δf_t represents the increase of estimated frequency of item t ($\forall t \in T$) from without attack to with attack. In MGA, the output item selection is performed according to the optimal solution maximizing the attacker's gain and sent to the server without perturbation.

Cao et al. describe the details of these three attacks against kRR, OUE, OLH in the frequency estimation and give theoretical analysis. The summary of the results is shown in Table 5.1. The table shows the overall gains of the three

	kRR	OUE	OLH
RPA (output-manipulation)	$\beta(\frac{r}{d} - f_T)$	$\beta(r - f_T)$	$-\beta f_T$
RIA (input-manipulation)	$\beta(1 - f_T)$	$\beta(1 - f_T)$	$\beta(1 - f_T)$
MGA (output-manipulation)	$\beta(1 - f_T) + \frac{\beta(d-r)}{e^\epsilon - 1}$	$\beta(2r - f_T) + \frac{2\beta r}{e^\epsilon - 1}$	$\beta(2r - f_T) + \frac{2\beta r}{e^\epsilon - 1}$

Table 5.1: MGA can achieve the highest **gains** against all three protocols. $\beta = \frac{M}{N+M}$ and $f_T = \sum_{t \in T} f_t$ in the table.

attacks against kRR, OUE, and OLH. MGA can achieve the highest gains for all protocols, clearly because MGA maximizes the gains. A notable point is a difference, summarized in Table 5.2, showing the difference in gains between MGA and RIA. They respectively correspond to *output-manipulation* and *input-manipulation* (described later) in this work. Note that the difference is remarkable, especially under the higher privacy budget.

Untargeted Attack. Albert et al. [38] analyze manipulation attacks in LDP. Compared to Cao et al.'s work, their study mainly focuses on *untargeted* attacks. The attackers aim to skew the original distribution and degrade the overall estimation accuracy of the server.

They suggest for the LDP protocols that the architecture is inherently vulnerable to malicious clients' manipulations. They suppose a general manipulation attack: the attacker injects M users in N clients in the LDP protocol. These injected users can send arbitrary data sampled from carefully skewed distributions to the server without supposed perturbation. We consider this attacker model corresponds to MGA in [39] and *output-manipulation* (described later) in this work. We should focus on one of their contributions: they show the general manipulation attack can skew the estimated distribution by $\Omega(\frac{M\sqrt{d}}{\epsilon N})$ in the frequency estimation, which causes more significant error than input-manipulation by about a $\frac{\sqrt{d}}{\epsilon}$ factor (Table 5.2). The difference is, for example, defined as l_1-norm of the original and skewed distribution.

Summary. We summarize these notable results in Table 5.2, showing how effective output-manipulation can attack compared to input-manipulation. The above two previous studies' common conclusion is highlighting the great necessity of enforcing the correctness of users' randomization to defend the output-manipulation attacks.

	kRR	OUE	OLH
Targeted Attack [39]	$+\left(\frac{\beta(d-r)}{e^\epsilon-1}\right)$	$+\left(\beta(2r-1)+\frac{2\beta r}{e^\epsilon-1}\right)$	$+\left(\beta(2r-1)+\frac{2\beta r}{e^\epsilon-1}\right)$
Untargeted Attack [38]		$\times\Omega\left(\frac{\sqrt{d}}{\epsilon}\right)$	

Table 5.2: Overall, output-manipulations are much more vulnerable than input-manipulation. The differences of both manipulations gain are calculated by output-manipulation gain − input-manipulation gain (resp. output-manipulation gain / input-manipulation gain) in Targeted (resp. Untargeted) Attack.

5.3 Problem Statements

Firstly, we give some notations to LDP protocols, partially following the above-mentioned in Section 5.2. We denote a single LDP protocol as π_i, where a client c_i sends sensitive data v to server S in ϵ-LDP manner. Encode and perturbation are denoted together as ϕ. ϕ is a probabilistic function (i.e., randomization mechanism) that takes $v \in [d]$ as input and output $y \in D$, such that output space $D = [d]$ if kRR, $D = \{0,1\}^d$ if OUE, $D = [g]$ if OLH. And we denote overall protocol including all clients as $\Pi = \{\pi_i | i \in [N]\}$.

5.3.1 Overview of the goal

An attacker against Π injects compromised users into the protocol to send many fake data to a central server. Note that such an attack results in manipulation against a single protocol π by each compromised user. Therefore, we consider security for π, and by protecting security for π, we can naturally protect security for Π. As for the attacker's capability, the attacker can access the implementation of ϕ because this is executed on clients' local, and he knows all parameters and functions including ϕ, ϵ, d, D and $Support(y)$, and employs this information to craft effective malicious outputs. However, in fact, there is little variation in the attacker's behavior because the server can easily deny the protocol if output $y \notin D$. Under such conditions, as shown in Figure 5.1, we can observe that an attacker can carry out the following two classes of attacks:

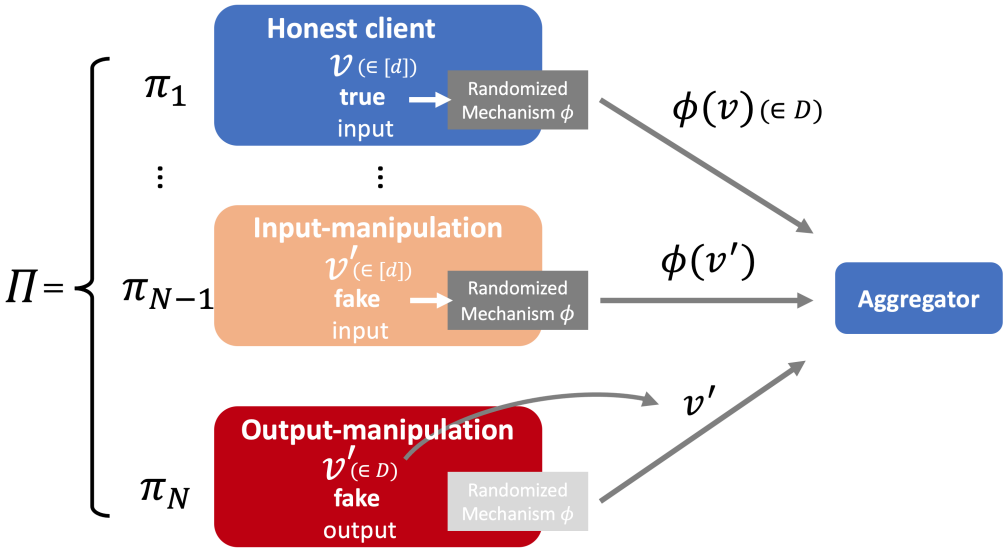


Figure 5.1: From top to bottom, normal protocol, input-manipulation attack and output-manipulation attack against an LDP protocol.

Input-manipulation supposes that the attacker can only select input data $v \in [d]$ and cannot interfere with other parameters and functions in π (middle in Figure 5.1). In other words, the attacker must send $y = \phi(v)$. But in a realistic setting, we should consider it a too strong assumption, as it allows an attacker to have complete control over the local system environment. For example, in targeted attack, RIA corresponds to this class of attacks.

Output-manipulation supposes the attacker can send arbitrary outputs to the server (bottom in Figure 5.1). This corresponds to the attacker can ignore all parameters and functions ϵ, ϕ by manipulating outputs directly. This attack is an entirely reasonable attack against a general LDP protocol because the server cannot distinguish between true data or fake data. In targeted attack, it corresponds to MGA or the attacks proposed in [38] for untargeted attack. Generally, this class effectively attacks, as shown in Table 5.2.

An important observation from Section 5.2.2 is that input-manipulation is much less effective than output-manipulation. Therefore, the natural direction is to defend against output-manipulation and limit the attack to the range of input-manipulation to achieve secure LDP protocols. On the other hand, it is hard to prevent input-manipulation completely. These have been studied in the fields of game theory [208] [209] or truth discovery [210], and we leave such a solution as

future work.

Overall, the goal is to mitigate attacks against the LDP protocols by completely defending output-manipulation and limiting to input-manipulation. For this purpose, we consider enforcing the correct mechanism ϕ for protocol π. The key idea is to make the protocol verifiable against malicious clients from a server. In the rest of this chapter, we refer to this property as *output-manipulation secure*. (It is also expressed simply as *secure*, and we call it as secure LDP protocol.)

Definition 11 (output-manipulation secure)**.** *An LDP protocol π is output-manipulation secure if any malicious client cannot perform output-manipulation and can only perform input-manipulation against π.*

5.3.2 Security definitions

In this subsection, we clarify what we should achieve for a secure LDP protocol. Similar to [206], security definitions of secure LDP protocol are consistent with a traditional secure two-party computation (2PC) protocol described in [211]. It considers an ideal world where we can employ Trusted Third Party (TTP) to execute arbitrary confidential computations indeed. And we aim to replace the TTP with a real-world implementation of cryptographic protocol $\pi = (c, S)$ between client c and server S. The protocol's flow when using a TTP is very simple. The client c sends input v to the TTP, and the TTP provides $y = \phi(v)$ to S. After all, c and S never receive any other information; S does not know v, and c does not know y. (S can estimate v from y and ϕ, but c's privacy should be guaranteed by LDP.)

While it is seemingly apparent that this ideal world's protocols will satisfy the requirements, let us review possible attacks closely. Goldreich [211] summarizes that there are just three types of attacks in malicious model 2PC against ideal world protocols; (1) denial of participation in the protocol; (2) fake input, not the true one; (3) aborting the protocol prematurely. We cannot hope to avoid these, but (1) and (3) cannot influence the estimation of original data distributions in LDP protocols. (2) is exactly input-manipulation described in the previous subsection. Thus, it is sufficient that the ideal world in 2PC is *output-manipulation secure* (see Definition 11) in the LDP protocols.

Considering the substituted cryptographic protocol $\pi = (c, S)$, let client c as prover $\mathcal{P}$ and server S as verifier $\mathcal{V}$ and $\pi = (\mathcal{P}, \mathcal{V})$. More specifically, we should

guarantee secure LDP under the worst case that both $\mathcal{P}$ and $\mathcal{V}$ behave maliciously. The case where $\mathcal{P}$ is malicious is obvious, considering output-manipulation. Still, in the $\mathcal{V}$'s case, it is because, given the original scenario of LDP, we need to guarantee the privacy of $\mathcal{P}$. And, we assume $\mathcal{P}$ is a polynomial computational adversary and $\mathcal{V}$ is unbounded.

Following [206], the ideal world protocol can be substituted with protocol π if (a) for any prover algorithm $\mathcal{P}^*$, $\mathcal{V}$ who receives $\phi(v) = y$ accepts only when $\mathcal{P}^*$'s secret input is surely v, or otherwise halts with negligible error; (b) for any prover algorithm $\mathcal{P}^*$, y is indistinguishable from other categories; (c) for any verifier algorithm $\mathcal{V}^*$, v is indistinguishable from other categories. Additionally, we need to verify that the randomization function ϕ used in the protocol does indeed satisfy ϵ-LDP.

Let $\text{view}_{\mathcal{P}}$ (resp. $\text{view}_{\mathcal{V}}$) as the set of messages generated by the protocol that $\mathcal{P}$ (resp. $\mathcal{V}$) can observe And let k as a security parameter that increases logarithmically with cryptographic strength. Then, the security definitions are reduced as following three properties:

- **Verifiability**: This property corresponds to the above-mentioned (a). We consider the protocol is verifiable if it satisfies as follows;

$$Pr[\mathcal{V} \text{ does not halts } |y \leftarrow \phi(*)] = 1 \text{ and,} \tag{5.6}$$

$$1 - Pr[\mathcal{V} \text{ halts } |y \leftarrow \mathcal{P}^*] < negl(k) \tag{5.7}$$

  where $negl(k)$ is negligible function in k, $y \leftarrow \phi(*)$ means y is obtained by correct execution of ϕ and $y \leftarrow \mathcal{P}^*$ means y is obtained by $\mathcal{P}^*$ other than correct $\phi(v)$.

- **Indistinguishability**: This property corresponds to (b) and (c). (b) satisfies if $\text{view}_{\mathcal{P}^*}$ has indistinguishable distributions for any input category $v \in [d]$. Formally, we define this property as follows; for any adversary $\mathcal{P}^*$,

$$|Pr[\mathcal{P}^*(\text{view}_{\mathcal{P}^*}, v) = y] - Pr[\mathcal{P}^*(v) = y]| < negl(k) \tag{5.8}$$

  where $negl(k)$ is negligible function in k. This means that a malicious client can use any information obtained from the protocol but only get negligible information about the final output of the server side. Similarly, (c) satisfies if, for any unbounded adversary $\mathcal{V}^*$,

$$|Pr[\mathcal{V}^*(\text{view}_{\mathcal{V}^*}, y) = v] - Pr[v|y]| < negl(k) \tag{5.9}$$

- **Local Differential Privacy**: The randomization mechanism ϕ in the given protocol must satisfy ϵ-LDP as shown in Definition 9. The verification of the correct execution is performed in Eq. (5.6).

5.4 Proposed Method

We design secure LDP protocols for kRR, OUE, and OLH, respectively, to completely defend against output-manipulations. In our method, a major building block is the Cryptographic Randomized Response Technique (CRRT) [206], which employs Pedersen's commitment scheme [212] for secure verifiability using the additive homomorphic property, and Naor-Pinkas 1-out-of-n Oblivious Transfer (OT) technique [65] for tricks for a verifiable randomization mechanism. Overall, the proof of validity is based on disjunctive proof [213]. It is a lightweight interactive proof protocol based on a secret sharing scheme and can perform witness-indistinguishable [214] proofs of knowledge (similar to zero-knowledge proofs). Combined with the security of the encryption scheme proposed in [206], it is possible to securely prove that the output value y is obtained by sampling from a probability distribution that satisfies the ϵ-LDP, i.e., $y = \phi(v)$. For simplicity, we explain several phases separately in the following protocol description (Protocol 1, 2), but they can be done simultaneously in the actual implementation.

Before explaining the protocols in detail, we introduce the following cryptographic setting. Assume that p and q are sufficiently large primes such that q divides $p-1$, $\mathcal{Z}_p$ has a unique subgroup G of order q. q is the shared security parameter between $\mathcal{P}$ and $\mathcal{V}$. Security parameter k is $k = \log_2 q_{max}$ such that q_{max} is the maximum value of possible q. We select g and h as a public key. They are two generators of G, and their mutual logarithms $\log_g h$ and $log_h g$ are hard to compute. We use this public key in the following protocols.

5.4.1 Secure kRR

Protocol 1 shows the details of the secure version of kRR, an extension of CRRT [206] to satisfy LDP for multidimensional data. As a whole, in the setup phase, both $\mathcal{P}$ and $\mathcal{V}$ prepare the same parameters l, n, z from accuracy parameter $width$ and privacy budget ϵ by Algorithm 15. l, n, z identify a categorical probability distribution that satisfies LDP, and we use it in 1-out-of-n OT for verifiable

Algorithm 15 DecideSharedParameters

Input: ϵ, $width$

$i \leftarrow \lfloor \frac{e^\epsilon}{(d-1)+e^\epsilon} \rfloor$ ▷ as an integer
while $i > 0$ **do**
  if $(width - i)$ divides $(d-1)$ **then**
    $g \leftarrow gcd(i, width, \frac{width-i}{d-1})$
    $l, n \leftarrow \frac{i}{g}, \frac{width}{g}$
    break
  end if
  $i \leftarrow i - 1$
end while
$z \leftarrow \max([l, \frac{n-l}{d-1}]) + 1$

Output: l, n, z

random sampling. In mechanism phase, $\mathcal{P}$ creates a vector $\boldsymbol{\mu}$ representing the categorical distribution containing n data where each data μ_i corresponds to one of the categories $[d]$. $width$ (i.e., n) is the size of the vector and decides a trade-off between accuracy to approximate LDP and overheads caused by the protocol. For proof P2, we use z^{μ_i} instead of μ_i. All z^{μ_i} is encrypted to y_i by an encryption scheme that combines Pedersen's commitment and OT. Only the μ_σ, where σ is pre-chosen by $\mathcal{V}$, can be decrypted correctly. Such a trick allows us to surely perform random sampling from vector $\boldsymbol{\mu}$ representing the categorical distribution. In the proof phase, two proofs are verified in the protocol. The first one is a disjunctive proof for each encrypted data y_i belonging to one of the categories $[d]$ (**P1**). The second one also uses a disjunctive proof that the summation of the vector used as a categorical distribution in the OT belongs to one of the possible values (**P2**). There are just d possible values for the summation of $\boldsymbol{\mu}$ (4.(a)).

Here, we confirm that Protocol 1 is secure. From the protocol, the prover and the verifier obtain $\text{view}_\mathcal{P} = \{g^a, g^b, g^{ab-\sigma-1}, x_i, x\}$ and $\text{view}_\mathcal{V} = \{w_i, y_i, com_i^{(j)}, c_i^{(j)}, h_i^{(j)}, com_j, c_i, h_i\}$ for all $i \in [n], j \in [d]$ respectively.

Firstly, we consider indistinguishability. The encryption scheme (e.g., μ_i is encrypted to y_i) is the same as the one presented in [206], which has been shown to be sufficiently indistinguishable for $\mathcal{P}^*$ and $\mathcal{V}^*$. That is, it is as hard for $\mathcal{P}^*$ to know about the σ, and also hard for $\mathcal{V}^*$ to guess the distribution of $\boldsymbol{\mu}$ and input v. Considering the attacker views, for $\mathcal{P}^*$, calculating σ from $\text{view}_\mathcal{P}$ is as hard

as the Decisional Diffie Hellman (DDH) problem. And x and x_i are completely random integers. For $\mathcal{V}^*$, (w_i, y_i) of $\text{view}_{\mathcal{V}}$ is indistinguishable by the security of the cryptographic scheme, and $(com_i^{(j)}, c_i^{(j)})$ is also indistinguishable because of the secret sharing scheme [213]. Verifiability is satisfied by proofs, P1 and P2. If both P1 and P2 are verified, $\mathcal{V}$ itself selects one value from the verified vector by OT. Then, for any operation by $\mathcal{P}^*$, $\mathcal{V}$ can confirm the correctness of the protocol. Hence, verifiability entirely depends on the protocol that proves the P1 and P2. We use disjunctive proofs and Eq.(5.6) and Eq.(5.7) are respectively satisfied by the completeness and soundness of the disjunctive proofs shown in [213]. Lastly, Algorithm 15 definitely generates l, n such that $\frac{l}{n} \leq \frac{e^\epsilon}{(d-1)+e^\epsilon}$ and $\frac{n-l}{d-1} \geq \frac{1}{(d-1)+e^\epsilon}$. Hence, because random sampling from $\boldsymbol{\mu}$ is equivalent to kRR with $p = \frac{l}{n}, q = \frac{n-l}{d-1}$, at least ϵ-LDP is satisfied.

Protocol 1 secure kRR

Client c as prover $\mathcal{P}$ who holds an secret input $v \in [d]$ and server S as verifier $\mathcal{V}$. ϵ is privacy budget and *width* is a parameter representing the degree of approximation.

1. **Setup phase.**
   (a) $\mathcal{P}$ and $\mathcal{V}$ run DecideSharedParameters$(\epsilon, width)$ and prepare l, n, z as shown in Algorithm 15. This is an algorithm for approximating integers l, n, z for given ϵ with as little degradation in accuracy as possible while still satisfying privacy protection.
   (b) $\mathcal{V}$ selects $\sigma \in [n]$. And $\mathcal{P}$ prepares a n-length random number vector $\boldsymbol{\mu} = (\mu_1, ..., \mu_n)$ where for all $1 \leq i \leq n$, $\mu_i \in [d]$, the vector satisfies $\#\{\mu_i | \mu_i \in \boldsymbol{\mu}$ and $\mu_i = v\} = l$ and for all $\{v' | v' \in [d] \setminus \{v\}\}$, $\#\{\mu_i | \mu_i = v'\} = \frac{n-l}{d-1}$ where $\#\{\cdot\}$ returns count on a set.
2. **Mechanism phase.**
   (a) $\mathcal{V}$ picks random $a, b \leftarrow \mathbb{Z}_q$ and sends g^a, g^b and $g^{ab-\sigma+1}$ to $\mathcal{P}$.
   (b) For all $i \in \{1, ..., n\}$, $\mathcal{P}$ performs the following subroutine; (1) Generate (r_i, s_i) at random; (2) Compute $w_i \leftarrow g^{r_i}(g^a)^{s_i} = g^{r_i + a s_i}$ and $h_i \leftarrow (g^b)^{r_i}(g^{ab-\sigma+1} g^{i-1})^{s_i} = g^{(r_i + a s_i)b + (i-\sigma)s_i}$; (3) Encrypt μ_i to y_i as $y_i \leftarrow g^{z^{\mu_i}} h^i$. Then, send (w_i, y_i) to $\mathcal{V}$.
   (c) $\mathcal{V}$ computes w_σ^b where σ is what $\mathcal{V}$ choose at setup phase, and computes $g^{\mu_\sigma} \leftarrow \frac{y_\sigma}{h^{w_\sigma^b}}$. And then, find μ_σ from the result and g. Thus, $\mathcal{V}$ receives μ_σ as a randomized output from $\mathcal{P}$.
3. **Proof phase for P1.**
   (a) For all $j \in [d] \setminus \{\mu_i\}$, for all $i \in \{1, ..., n\}$, $\mathcal{P}$ generates challenge $c_i^{(j)}$ and response $s_i^{(j)}$ from $\mathbb{Z}_q$ and prepares commitments $com_i^{(j)} \leftarrow h^{s_i^{(j)}} / (y_i / g^{z^j})^{c_i^{(j)}}$. For $\{\mu_i\}$ and for all $i \in \{1, ..., n\}$, $\mathcal{P}$ generates $w_i \leftarrow \mathbb{Z}_q$ and let $com_i^{(\mu_i)} = h^{w_i}$. Then, send $com_i^{(j)}$ to $\mathcal{V}$, for all i, j.
   (b) $\mathcal{V}$ picks $x_i \leftarrow \mathbb{Z}_q$ for all $i \in \{1, ..., n\}$ and sends it to $\mathcal{P}$.
   (c) For all $i \in \{1, ..., n\}$, $\mathcal{P}$ computes $c_i^{(\mu_i)} = x_i - \sum_{j \in [d] \setminus \mu_i} c_i^{(j)}$ and $s_i^{(\mu_i)} = v_i c_i^{(\mu_i)} + w_i$. Then, send $c_i^{(j)}$ and $s_i^{(j)}$ for all i, j to $\mathcal{V}$.
   (d) Finally, $\mathcal{V}$ checks if $h^{s_i^{(j)}} = b(y_i / g^{z^j})^{c_i^{(j)}}$ for all $j \in [d]$ and $x_i = \sum_{j \in [d]} c_i^{(j)}$, for all $i \in 1, ..., n$. Otherwise halts.
4. **Proof phase for P2.**
   (a) For all $j \in [d] \setminus \{v\}$, $\mathcal{P}$ generates challenge c_j and response s_j from $\mathbb{Z}_q$ and prepares commitments $com_j \leftarrow h^{s_j} / (\prod_{i \in \{1,..,n\}} y_i / g^{Z_j})^{c_j}$ where $Z_j = \frac{n-l}{d-1} \left(\sum_{k \in [d] \setminus \{j\}} z^k \right) + l z^j$. And $\mathcal{P}$ generates $w \leftarrow \mathbb{Z}_q$ and let $com_v = h^w$. Then, send com_j to $\mathcal{V}$, for all $j \in [d]$.
   (b) $\mathcal{V}$ picks $x \leftarrow \mathbb{Z}_q$ and sends it to $\mathcal{P}$.
   (c) $\mathcal{P}$ computes $c_v = x - \sum_{j \in [d] \setminus \{v\}} c_j$ and $s_v = \left(\sum_{i \in 1, ..., n} v_i \right) c_v + w$. Then, send c_j and s_j for all j to $\mathcal{V}$.
   (d) Finally, $\mathcal{V}$ checks if $h^{s_j} = b(\prod_{i \in \{1,..,n\}} y_i / g^{Z_j})^{c_j}$ for all $j \in [d]$ and $x = \sum_{j \in [d]} c_j$. Otherwise halts.

5.4.2 Secure OUE

We show the secure version of the OUE protocol in Protocol 2. Unlike kRR, OUE sends a d-length bit vector where each i-th bit corresponds to the likelihood that the client has the item $i \in [d]$. In OUE, mechanism ϕ performs random bit flips with given constant probability independently for each bit. The Bernoulli distributions, which determine the probabilities of each flip, are approximated by a distribution of n-length bit vectors. As in the case of kRR, verifiable random sampling is achieved by a trick using Pedersen's commitment and OT. However, there are d distribution vectors since it needs for each category.

Protocol 2 secure OUE

$\mathcal{P}$, $v \in [d]$, $\mathcal{V}$, $width$, ϵ as with Protocol 1.

1. **Setup phase.**
   (a) $\mathcal{P}$ and $\mathcal{V}$ set l, n as $\lceil \frac{1}{1+e^\epsilon} \cdot width \rceil$ and $width$ itself respectively.
   (b) $\mathcal{V}$ selects d random numbers $\boldsymbol{\sigma} = \{\sigma_1, ..., \sigma_d\}$ where $1 \leq \sigma_j \leq n$. $\mathcal{P}$ prepares d n-length random bit vectors $\vec{\boldsymbol{\mu}} = (\boldsymbol{\mu_1}, ..., \boldsymbol{\mu_n})$ such that $\boldsymbol{\mu_j} = (\mu_1^{(j)}, ..., \mu_n^{(j)})$ where all $\mu_i^{(j)} \in \{0, 1\}$, and the vector satisfies $\sum_i \mu_i^{(j)} = n - l$ if $j = v$ and $\sum_i \mu_i^{(j)} = l$ if $j \neq v$.
2. **Mechanism phase.**
   (a) $\mathcal{V}$ picks random $a_j, b_j \leftarrow \mathbb{Z}_q$ and sends g^{a_j}, g^{b_j} and $g^{a_j b_j - \sigma_j + 1}$ to $\mathcal{P}$ for all $j \in [d]$.
   (b) For all $j \in [d]$ and $i \in \{1, ..., n\}$, $\mathcal{P}$ performs the following subroutine; (1) Generate $(r_i^{(j)}, s_i^{(j)})$ at random; (2) Compute $w_i^{(j)} \leftarrow g^{r_i^{(j)}} (g^{a_j})^{s_i^{(j)}} = g^{r_i^{(j)} + a s_i^{(j)}}$ and $h_i^{(j)} \leftarrow (g^{b_j})^{r_i^{(j)}} (g^{a_j b_j - \sigma_j + 1} g^{i-1})^{s_i^{(j)}} = g^{(r_i^{(j)} + a s_i^{(j)}) b_j + (i - \sigma_j) s_i^{(j)}}$; (3) Encrypt $\mu_i^{(j)}$ to $y_i^{(j)}$ as $y_i^{(j)} \leftarrow g^{\mu_i^{(j)}} h^{h_i^{(j)}}$. Then, $\mathcal{P}$ sends all $(w_i^{(j)}, y_i^{(j)})$ to $\mathcal{V}$.
   (c) For all $j \in [d]$, $\mathcal{V}$ computes $g^{\mu_{\sigma_j}^{(j)}} \leftarrow y_{\sigma_j}^{(j)} / h^{(w_{\sigma_j}^{(j)})^{b_j}}$. And then, find μ_{σ_j}. Thus, $\mathcal{V}$ receives $[\mu_{\sigma_1}, ..., \mu_{\sigma_d}]$ as a randomized output from $\mathcal{P}$.
3. **Proof phase for P1.**
   (a) For all $j \in [d]$, for all $i \in \{1, ..., n\}$, $\mathcal{P}$ generates challenge $c^{(j)}_{1-\mu_i^{(j)}, i}$ and response $s^{(j)}_{1-\mu_i^{(j)}, i}$ from $\mathbb{Z}_q$ and prepares commitments $com^{(j)}_{1-\mu_i^{(j)}, i} \leftarrow h^{s^{(j)}_{1-\mu_i^{(j)}, i}} / (y_i^{(j)} / g^{\mu_i^{(j)}})^{c_i^{(j)}}$. Generate $w_i^{(j)} \leftarrow \mathbb{Z}_q$ and compute $com^{(j)}_{(\mu_i^{(j)}), i} \leftarrow h^{w_i^{(j)}}$. Then, send $com^{(j)}_{\{0,1\}, i}$ to $\mathcal{V}$, for all i, j.
   (b) $\mathcal{V}$ picks $x_i^{(j)} \leftarrow \mathbb{Z}_q$ for all $j \in [d]$ and $i \in 1, ..., n$ and sends it to $\mathcal{P}$.
   (c) For all $j \in [d]$ and $i \in \{1, ..., n\}$, $\mathcal{P}$ computes $c^{(j)}_{\mu_i^{(j)}, i} = x_i^{(j)} - c^{(j)}_{1-\mu_i^{(j)}, i}$ and $s^{(j)}_{\mu_i^{(j)}, i} = v_i^{(j)} c^{(j)}_{\mu_i^{(j)}, i} + w_i^{(j)}$. Then, send $c^{(j)}_{\{0,1\}, i}$ and $s^{(j)}_{\{0,1\}, i}$ for all i, j to $\mathcal{V}$.
   (d) Finally, $\mathcal{V}$ checks if $h^{s^{(j)}_{\{0,1\}, i}} = b (y_i^{(j)} / g^{\{0,1\}})^{c^{(j)}_{\{0,1\}, i}}$ and $x_i^{(j)} = c_{0,i}^{(j)} + c_{1,i}^{(j)}$, for all $i \in \{1, ..., n\}$ and for all $j \in [d]$. Otherwise halts.
4. **Proof phase for P2. (Simplified because it is similar to P1.)**
   (a) $\mathcal{P}$ generates and sends all $com^{(j)}_{\{p,q\}}$ to $\mathcal{V}$.
   (b) $\mathcal{V}$ picks $x_j \leftarrow \mathbb{Z}_q$ for all $j \in [d]$ and sends it to $\mathcal{P}$.
   (c) $\mathcal{P}$ sends $c^{(j)}_{\{p,q\}}$ and $s^{(j)}_{\{p,q\}}$ for all j to $\mathcal{V}$
   (d) $\mathcal{V}$ checks if $h^{s_p^{(j)}} = b (\prod_{i \in 1, .., n} y_i^{(j)} / g^{n/2})^{c_p^{(j)}}$ and $h^{s_q^{(j)}} = b (\prod_{i \in 1, .., n} y_i^{(j)} / g^{l})^{c_q^{(j)}}$ and $x_j = c_p^{(j)} + c_q^{(j)}$ for all $j \in [d]$. Otherwise halts.
5. **Proof phase for P3.**
   (a) $\mathcal{P}$ computes $h_{sum} \leftarrow \sum_{i,j} h_i^{(j)}$ and sends h_{sum} to $\mathcal{V}$.
   (b) $\mathcal{V}$ checks if $h^{h_{sum}} g^{n/2 + l(d-1)} = \prod_{i,j} y_i^{(j)}$. Otherwise halts.

In addition, each vector's distribution is one of two types: the j-th vector such that the secret input $v = j$ or otherwise (i.e., p or q in Eq. (5.4)). Thus, we perform independent OT and decide 0 or 1 for d categories and finally, get the randomized output $[\mu_{\sigma_1}, ..., \mu_{\sigma_d}]$.

Then, similar to secure kRR, we must show that all Bernoulli distributions represented by d vectors are correct. Specifically, the proofs are that all elements of bit vectors $\vec{\boldsymbol{\mu}}$ are surely a bit (0 or 1) (**P1**) and the distribution of the vectors are surely equivalent to either of p or q of Eq. (5.4) (**P2**). The number of p and q are 1 and $d-1$ respectively (**P3**). If all these three proofs are verified, we can confirm the OUE protocol is simulated correctly. Like kRR's proofs, **P1** and **P2** are proved by d disjunctive proofs. **P3** is based on the hardness of the discrete logarithm problem. $\mathcal{P}$ cannot find h_{sum} in polynomial time without all correct $h_i^{(j)}$ that is used when encrypting $y_i^{(j)}$. While $\mathcal{P}$ has to release h_{sum}, this is information theoretically indistinguishable from $\mathcal{V}$ for each $h_i^{(j)}$ unless $n = d = 1$. Security statements for the secure OUE protocol are similar to secure kRR. For LDP, as we can see 1.(a) in Protocol 2, we set $q = l/n$ such that $\frac{l}{n} \geq \frac{1}{1+e^{\epsilon}}$.

5.4.3 Secure OLH

To make OLH output-manipulation secure, basically, we can use Protocol 1 except that it requires sharing of a hash function and using reduced output category space. As a first step, $\mathcal{V}$ generates and sends a seed s to $\mathcal{P}$ to initialize hash function $H_s : v \to v'$ where $v \in [d]$ and $v' \in [g]$. $\mathcal{V}$ and $\mathcal{P}$ use the same H_s as a hash function. We can apply Protocol 1 to achieve secure OLH by using category set $[g]$ instead of $[d]$ and sensitive input value v is handled as $v' = H_s(v)$. The rest of the steps are almost the same as kRR.

Even if $\mathcal{P}^*$, who does not use the hash function correctly, participates in the protocol, $\mathcal{V}$ can easily detect it if it sends the output of a different output space, i.e. $y \notin [g]$. If the attacker does not use a different output space, the attack can only be equivalent to input-manipulation because $\mathcal{V}$ verifies the correctness of the categorical distribution used in random sampling after applying the hash function.

5.5 Evaluation

In this section, we evaluate and analyze the performance of the proposed protocols. The code in Python is available on GitHub[b].

Experimental setup. We use an HP Z2 SFF G4 Workstation, with a 4-core 3.80 GHz Intel Xeon E-2174G CPU (8 threads, with 8MB cache), and 64GB RAM. The host OS is Ubuntu 18.04 LTS. The client and server exchange byte data serialized by pickle (from Python standard library) over TCP. We use $\epsilon = 1.0$ and in OLH, set $g = d/2$ as the hashed space instead of $g = \lfloor e^{\epsilon}+1 \rfloor$ for demonstration.

Parameter generator. First, we analyze the approximated probability distribution generated by the proposed method. In the secure kRR protocol, we approximate the probability distribution where we generate data to satisfy LDP by Algorithm 15. Figure 5.2 shows how accurate the algorithm generates discrete distribution for $\epsilon = (0, 5]$ and for $width = \{100, 1000\}$. The red curve represents probability p for the normal mechanism, and the blue one represents the approximated one. When the $width$ is small, there is a noticeable loss of accuracy due to approximation. However, with a sufficiently large $width$, the approximated p has a sufficiently small loss. As the $width$ increases, the performance degrades, indicating that there is a trade-off between the accuracy of the probability approximation and the performance. This is true not only for kRR but also for OUE and OLH. For secure OUE, in the right-side of Figure 5.2, we compare probability q because p is constant in OUE. It is almost an exact discrete approximation with a small $width$. This is due to the difference in the structure of the vectors that form the probability distribution, with OUE having a simpler structure.

Performance. We evaluate the performances of our proposed method. Figure 5.3 shows the total bandwidths caused by communications of the entire protocol for each of the three methods with different category sizes. Generally, as the category size increases, the total bandwidth also increases. While it increases linearly in OUE, there are fluctuations in kRR and OLH. This is because the probability value that Algorithm 15 approximates may have a smaller denominator (i.e., n) by reduction, which can make the distribution vector smaller. Overall, a larger $width$ generates almost linear increases in bandwidth. And for the same $width$, secure OUE causes a larger communication overhead than the others. However, as mentioned in the previous paragraph, secure OUE can approximate the proba-

[b] `https://github.com/FumiyukiKato/verifiable-ldp`

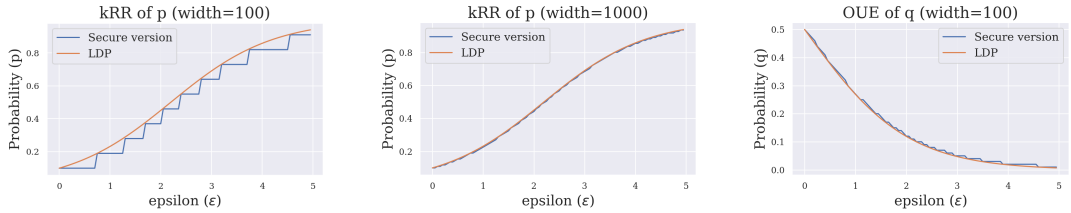


Figure 5.2: In secure kRR, with a sufficiently large *width*, categorical distribution by Algorithm 15 can accurately approximate the LDP distributions (left and middle). In secure OUE, it is almost exact discrete approximation with relatively small *width*. (right)

bility distribution with high accuracy using a smaller *width*. Hence, in particular, when the number of categories is large, secure OUE is considered to be more efficient by using a smaller *width*. Figure 5.3 shows that, comparing kRR with $width = 1000$ and OUE with $width = 100$, many categories require several times more bandwidth. On the other hand, when the discretized probability distribution can be approximated with a small denominator by reduction, kRR and OLH show a very small bandwidth. When comparing kRR and OLH, OLH is smaller overall. This is due to the fact that the output space is reduced by hashing.

Figure 5.4 shows the total execution time from the time the client sends the first request until the entire protocol is completed. Most of the characteristics are similar to those of bandwidth. As the size of the proofs that need to be computed increases, the execution time is also expected to increase. The only difference is in OLH, which takes extra time to execute the hash function. However, as the number of categories becomes larger, the influence becomes smaller.

Therefore, the overhead can be minimized by providing an optimal efficiency privacy budget for kRR and OLH, and by using different methods for different *widths*. The overhead is expected to increase as the number of categories increases, but since the limit on the number of categories is determined to some extent by the use of LDP, we do not think this is a major problem.

At the end, impressively, our method is algorithm-only, making it more feasible than alternatives that assume secure hardware providing TEE [62, 70]. Nevertheless, overall, we believe the overhead is acceptable. This is due to the fact that we use relatively lightweight OT techniques as a building block.

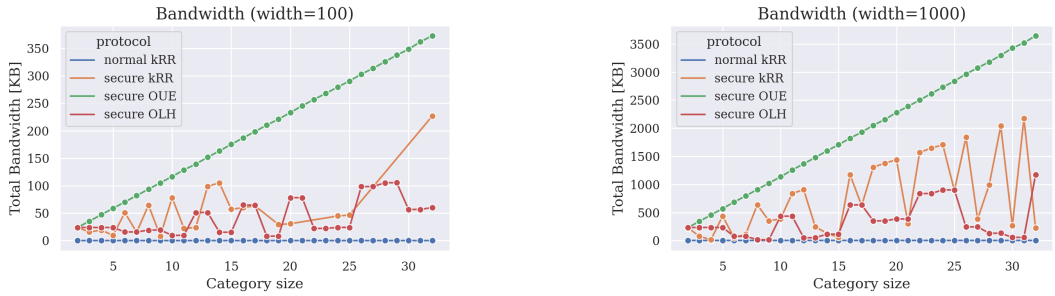


Figure 5.3: With the same *width*, the communication costs of kRR and OLH are small. However, OUE can approximate LDP accurately with small widths (Figure 5.2).

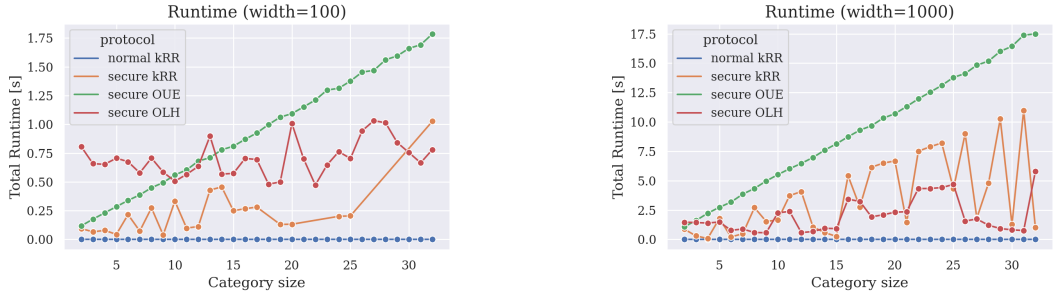


Figure 5.4: The characteristics of runtime is similar to bandwidth. OLH takes a little longer because of the hashing.

5.6 Conclusion

In this work, we showed how to prevent malicious clients from attacking LDP protocols. An important observation was the effectiveness of output-manipulation and the importance of protection against it. Our approach was a verifiable randomization mechanism satisfying LDP. Data collectors can verify the completeness of executing an agreed randomization mechanism for every possibly malicious data provider. The proposed method was based on only lightweight cryptography. Hence, we believe it has high feasibility and can be implemented in various and practical data collection scenarios.

5.6.1 Future works

- As this work shows, client-side security in FL and/or FA is still in the early stage and is a very difficult research topic. This study focuses on a simple FA task. On the other hand, the studied principles of output-manipulation and input-manipulation are more generally important. In the context of FL, these are also known as *model update poisoning* and *data poisoning* attacks [93]. And it is suggested that the latter has a greater capacity for attack [37]. Therefore, extending our defense method, i.e., client-side verifiability, to more complex operations, i.e., model optimization calculations, is an important direction. The LDP protocol itself, which was our target, has also received attention since our research, and improvements using zero-knowledge proof (ZKP) [215] and blockchain [216] have been proposed.

  The important point is that relatively lightweight cryptographic techniques that were easily applicable to simple LDP protocols are not easily applicable to complex computations such as ML optimization algorithms. Therefore, it is necessary to consider efficient ways to prevent output-manipulation by utilizing TEE on the client side as in [217] or by combining more practical ZKP techniques [218]. It is not clear that all calculations need to be verifiable, and perhaps partial verifiability may be sufficient to prevent the effective attack.

- Another direction is to prevent input-manipulation. This could be partially addressed, for example, by using ZKP to verify that the input data is within the expected range. However, it seems to be almost impossible to verify that the client providing the data is providing truthful data. However, it has been addressed partially by truth discovery-like approaches, evaluating the shared information from clients [219, 220] or blockchain-based approaches, verifying the integrity of training data [221, 222]. For example, Zhang et al., [220] propose to verify the input data quality by observing model losses for validation data. Furthermore, it may be necessary to propose a comprehensive framework, combining various aspects including designing incentives for FLs to participate [223].

CHAPTER 6

Discussion

We have described our proposed frameworks ULDP-FL, OLIVE and VLDP in Chapters 3, 4, and 5, respectively. In this chapter, we aim to explore the potential profound social impact of our comprehensive studies produced in this thesis, which is committed to addressing and resolving critical, previously unsolved challenges in private and secure FL. Note that the future works from a technical point of view on each topic are summarized at the end of each chapter.

6.1 Social Impact

We believe that our research will have a significant impact on society in terms of promoting the implementation of privacy-preserving FL in practice.

In general, current privacy-preserving data analysis remains at the trial stage, although the growing sense of social responsibility is increasingly compelling companies to implement these privacy-preserving techniques. In particular, the increasing number of privacy laws and regulations, such as GDPR, are forcing companies to implement privacy protections. Also, it would enhance the social credibility of companies. Nevertheless, privacy-preserving data analysis has not gone beyond the trial stage. We believe that the bottleneck in bringing current privacy-preserving technology into production-level is (1) the lack of sufficient privacy protection and (2) the fact that the architecture for privacy protection

conversely introduces additional risks.

(1) Of course, the lack of sufficient privacy protection is not an easy problem. In today's society, it is probably impossible for anyone to answer what privacy protection can be claimed to be sufficient. For companies, the risk of insufficient privacy protection is the unfortunate consequence of just sacrificing utility and efficiency for a system that fails to provide a level of privacy guarantee deemed valuable. Even if we consider formal privacy guarantees such as DP, these guarantees are often limited, for example, when they actually provide just record-level DP guarantee or too large privacy budget and so on. In such cases, companies often refrain from loudly proclaiming the value of the privacy protection and the technologies. As a result, most of current privacy-preserving technologies perpetually remain limited to trial usage or are used solely as a corporate showcase. We believe this approach is often adopted because it allows companies to avoid rigorous discussions about strict privacy guarantees. In other words, we would point out that incomplete or non-confident (for the company) privacy protection is almost worthless and never being used in production-level applications.

(2) Privacy protection architecture often necessitates a distributed system like FL, which can introduce new security risks as described in Chapter 1. This situation leads to a dynamic where companies are hesitant to implement privacy-preserving technologies due to the potential for the additional risks. Furthermore, as mentioned in (1), if the level of privacy protection cannot be assured as sufficient, it will ultimately face rejection by companies. We believe while a company may accept a slight decrease in utility or efficiency instead of privacy protection, it is unlikely to readily accept such an increase in security risk.

This doctoral research makes a fundamental contribution to addressing the challenge of privacy protection technologies that often remain limited to the trial stage. We discuss the respective social impacts of our proposed three frameworks.

ULDP-FL: Cross-silo DP-FL has been proposed and studied, aiming at various applications in industries or medical fields. However, given the record-level DP guarantee, its justification for privacy protection is clearly insufficient, and it will likely end up remaining in the trial stage. With our proposed ULDP-FL, the privacy protection that can be claimed becomes significantly more acceptable because the unit of privacy should always be a user, not a record. This is particularly critical when companies need to justify their privacy practices to consumers. The use case where a single user exists across silos is quite common,

therefore, the social impact is big enough for practitioners to apply DP-FL in production, at least from the privacy definition perspective. Many companies in the same domain will be able to train private models in production-level applications, even if they have overlapped users. The same is true for hospitals that have overlapped patients. Finally, let us emphasize that this was impossible without the ULDP-FL framework.

OLIVE: TEE is a powerful security technology with potential significant impact on existing systems. However, its use in real-world applications becomes hard if clear vulnerabilities in TEE lead to privacy leakage. The situation where a TEE is used for privacy protection but ends up being a source of privacy leakage is highly problematic. This inability of companies to guarantee complete (or at least confident) privacy likely keeps the technology confined to the trial stage. Our proposed OLIVE framework is the first to address this issue in FL by providing a solution for privacy leakage via TEE's vulnerability. It fully justifies the true privacy protection provided by TEE. This development enhances the social credibility of privacy-preserving FL technology using TEE, and companies will be able to claim that they can clearly have confidence in the privacy protection beyond the trial stage. Moreover, FL using TEE is quite superior in terms of the ease of understanding of privacy protection for consumers. Remarkably, this approach is applicable to almost all FL applications and we believe it will be of great social significance to make this feasible.

VLDP: The existence of malicious clients in Federated Analytics creates a dynamic that drives companies away from deploying privacy-preserving new architectures. This indicates that the new privacy-preserving architecture, intended as a defense mechanism, paradoxically introduces new vulnerabilities. Our proposed VLDP is one of the solutions for such a dynamic, which eliminates the side-effect security risks associated with deploying new privacy protection technologies. OLIVE will be a solution against a malicious server as well. We believe that it will have the social impact of removing the negative aspects of privacy protection technologies to implement. Such security technologies are crucial yet often overlooked. Our work pioneered this direction and will promote the successive research for facilitating social implementation of private and secure FL.

Overall, our research significantly advances the practical application of privacy-preserving FL, which always has been in the trial stage because of its clear imperfection of privacy guarantee and additional security risks. In particular, they

will lead the entity executing the FL to declare with confidence that it truly provides privacy protection, thereby increasing confidence in the sufficiency claim of privacy protection, which is one of the most important factors in production-level private data analysis. Additionally, our research aids in the social adoption of such new privacy-preserving architectures by addressing potential security risks.

6.2 Real-World FL Application

More specifically, we explore the practical application of the proposed frameworks to real-world use cases. This includes assessing their necessity and applicability across various real-world FL applications. We will follow the classification of FL applications proposed by Li et al. [224], namely: Applications for mobile devices, Applications in industrial engineering, and Applications in Healthcare.

6.2.1 Application for Mobile Devices

The most typical application for which FL has been tested in industry so far is the prediction of smartphone keyboard input completion [30, 17, 109], including stickers [15] and emojis [31]. It has already been deployed in several real-world applications [17, 14, 16], some of which have announced that they guarantee DP [109, 15, 13]. Recently, FL has been used in smart home IoT devices [225, 226, 227] that collect physical information from the end user's living space and respond with AI.

These are typical cross-device FL settings, and at first glance, it seems irrelevant for ULDP-FL, whose primary target is cross-silo FL. However, it is conceivable that even in a cross-device FL setting, different devices may have data for a common user. For example, if there are multiple IoT devices that individually participate in FL in a single user's living space, user-level privacy needs to be guaranteed across those devices. Also, even if it is a mobile phone, the data possessed by an individual device is not restricted to that of a single user when data tied to multiple users, such as group chats, is used for training. Thus, even in cross-device FL, there is not necessarily a one-to-one correspondence between devices and individual data. For this situation, ULDP-FL can provide a necessary solution. Since the cross-device setting can be regarded as a special case of cross-silo, our proposed algorithm works correctly and effectively. Additionally,

Olive frees practitioners from the complex and constraining pairwise masking-based secure aggregation such as [66] for these systems, allowing for a simpler approach using TEE. It is also expected that the presence of Olive will allow careful users to participate in FL with confidence, even for applications that handle very critical information, such as home IoT devices. Also, since these mobile device-based applications will be deployed on a large scale, the risk of contamination by malicious participants is also expected to be relatively high. This may motivate companies to hesitate to deploy FL systems. A Vldp-like protection method will help dispel these concerns. It should also be noted that the introduction of TEE by Olive also allows for another line of defense mechanisms such as discovering malicious parameters, which is difficult to do with conventional secure aggregation. Overall, our research will definitely facilitate the implementation of privacy-preserving FL in production environments for these applications.

6.2.2 Application in Industrial Engineering

FL has the potential to promote integrated data utilization when data sharing is required between multiple corporate organizations, where data sharing is restricted by law or regulation [9, 228, 229, 230, 231]. Typical applications include training a model for detecting credit card fraud using transaction histories of data from multiple banks [9], and training a model for performing a visual inspection task in collaboration with multiple manufacturers [230].

This is exactly the scenario that Uldp-FL is mainly targeting. In the past, only incomplete privacy guarantees could be provided and therefore, it was far away from being used in production when collaborating with personal data from multiple companies. However, now it is possible to provide meaningful privacy guarantees with Uldp-FL. This is expected to reduce the number of companies hesitant to adopt FL, and will help FL flourish in this field. Banks and factories have a lot of their own private data including personal data, so facilitating these collaborations will be of great benefit to the whole.

In such business-to-business collaboration, the assumption that the server is trusted can often be valid. However, companies generally have large amounts of personal data or other stakeholders who are potentially at risk for privacy leaks. Therefore, Olive with TEE will have some value in such scenarios, which make it easy to obtain permission from clients for using their data within TEE. Similarly,

the risk of malicious clients may not be high, since silo is also assumed to be trusted with strong contracts and so on. The risk against a malicious client may be higher for critical models, such as image recognition models for automated driving [231] or models used in large-scale production lines [229], etc. VLDP will help to dispel these concerns.

6.2.3 Application in Healthcare

The medical domain is one of the most active areas where FL has been studied, and the main application is to train disease prediction models by combining patient data from different hospitals [106, 232, 233, 10, 234] and even different countries [11, 12]. Since the in-training and trained models may be exposed to users (e.g., other hospitals) for subsequent use, the need to guarantee DP has been argued to protect the privacy of patient data [232, 233, 108].

In general, hospital organizations are so socially trusted that patients entrust raw medical record data to the hospital and are not expected to commit fraud. However, it would be important to consider when learning medical data in conjunction with other commercial companies. For example, when training a model in which medical data is used by an insurance agent to derive reasonable pricing, it may be necessary to require to guarantee privacy protection for the training data, because patients do not trust the insurance company. Then, ULDP-FL is obviously important and should be integrated, since the patient may be seen in more than one hospital. We would like to emphasize that our proposed method is at least an indispensable technique to make such collaboration possible.

OLIVE can be applied if necessary. Although there would still be legal challenges, TEE is expected to work like a virtual *secure data room* which is an isolated physical room for data analysts often used in the medical domain to analyze sensitive data while prohibiting the data from leaving the room. Therefore, this secure computation model might be easily accepted by the existing medical domain. It is important to note that in FL applications, this is only possible with our proposed method. A malicious client can be very critical because it could change the predictive results of the disease. We believe this will be one of the major factors in the acceptance of DP-FL in the medical community where integrity is required. VLDP works as a defense if some hospitals are malicious, though this may not likely happen.

6.3 Limitation

Finally, we conclude this chapter by discussing the limitations of this thesis and some areas beyond its scope. The scope covered by the considered privacy/security properties, **A** - **E** (introduced in Section 1.4), in this thesis should also be discussed. Originally, these properties are derived from an analysis of sufficient existing FL studies. They cover a wide range of risk types because DP and TEE do not assume only specific attacks but are fairly general methods of defense. However, there can be another type of risks. [235] creates the latest version of a taxonomy for privacy risks from an AI perspective, which includes 12 high-level privacy risks. It is difficult to pick out FL-specific issues from most of them, but among those not covered in this thesis is, for instance, *Exclusion*, i.e., not informing the data owner of all uses of the data. The distributed and complicated system of FL may exacerbate the process of notifying users of their data use. Overall, however, we believe that all high-profile attacks in FL could be covered in this thesis.

Also note that the definition of FL could be a bit more generalized, including vertical FL and Split Learning [93]. The FL scenarios we cover throughout the thesis are only horizontal FL with the assumption that each client has the same data format. Basically, in the case of a single server and several distributed clients, our category of privacy/security properties seem to cover the attack surface well, since the assumed trust model is almost the same. However, detailed attack techniques may differ. For example, in vertical FL and Split Learning, it is necessary to exchange more than simple model deltas as we assume in this thesis; it is also necessary to exchange information such as table formats and intermediate model parameters called smashed data. These may cause another type of privacy/security risks that we do not cover.

CHAPTER 7

Concluding Remarks

In this thesis, to answer the research question: *"To enhance privacy and security in Federated Learning, how can Differential Privacy and Trusted Execution Environment be effectively integrated into Federated Learning?"*, we enumerate five key privacy/security properties in FL (described in Table 1.2 in Section 1.4) and studied three federated frameworks to complement them.

Chapter 2 provided comprehensive literature reviews of DP and TEE, respectively. They include our independent studies about DP and TEE described in detail in Section A and B in Appendix, respectively. The prior focused on differentially private counting query release on a high-dimensional dataset, and the latter targeted private contact tracing using trajectory data in TEE.

In Chapter 3, we examined rigorous privacy protection for models trained in FL with DP. In particular, we targeted the general setting of cross-silo FL, where each participating client corresponds to a certain size of institutions, with user-level DP guarantees. Under this setting, we showed that existing algorithms can only achieve impractical privacy guarantees and provided algorithms that achieve better privacy-utility tradeoffs. The proposed method directly guarantees user-level DP by applying per-user weighted clipping to the existing de facto DP-FedAVG. Furthermore, we proposed a utility-boosting weighting method and developed an MPC protocol to achieve it under a more stringent trust model.

In Chapter 4, we focused on the server-side TEE in FL, which enables guar-

anteeing the privacy of the shared gradients to the central untrusted server and providing better utility of DP-FL. Through analysis of memory access patterns for FL aggregation operations, we discovered the possibility of privacy risks when sparsified gradients are used. Using the observable memory access pattern information, we designed a novel attack that reveals private data and showed the effectiveness through experiments using real-world data. To defend against this attack, we proposed an oblivious algorithm such that the memory access patterns resulting from FL aggregation operations are independent of the input data. Finally, we evaluated the proposed Oblivious algorithm on a real data scale and showed its efficiency.

In Chapter 5, we focused on malicious clients and, in particular, proposed a defense against clients that deviate from the LDP protocol. This research direction was just beginning, and our focused task was a simple Federated Analytics, i.e., frequency estimation with guaranteed LDP. Because LDP requires perturbation of data on the client side, the central server does not have complete control over this protocol, allowing a malicious client to control the estimates. We showed that the attack can be partially prevented by developing a verifiable LDP protocol. We believe this type of client-side verifiability that we proposed in this method can be extended to prevent attacks in more complicated federated tasks including FL in the future.

Lastly, in Chapter 6, we discussed the profound social impact of the comprehensive studies produced in this thesis. We argued the specific benefits in the fields where real-world FL applications have been studied. Also, we described the limitations which are not covered in our proposed frameworks.

We believe that our research has made a significant contribution to the realization of private and secure FL and has convincingly argued for important directions in which to focus our efforts.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Professor Masatoshi Yoshikawa and Associate Professor Yang Cao, for their comprehensive guidance and support over the past six years since my undergraduate days. Their mentorship in the art of academic research has been invaluable for me. I am proud to have spent my doctoral studies under such honorable and brilliant supervisors. Special thanks to Professor Li Xiong, my research supervisor at Emory University. Her exceptional mentorship has significantly contributed to my dissertation and she has been and will be a great role model for me in my future life. I am also grateful to Associate Professor Makoto P. Kato, my undergraduate supervisor, who transformed my perception of academia into something truly inspiring. I thank my research advisor, Professor Tomohiro Kuroda, for his valuable insights, from his medical informatics class to my doctoral research, especially on the application of technology in medicine. My gratitude goes to Professors Takayuki Ito and Yasuo Okabe for their roles as examiners of my thesis. I am extremely thankful to Doctor Tsubasa Takahashi for instilling in me the methodology of academic research during my internship at LINE Corporation. His mentorship was pivotal in shaping my academic achievements. My appreciation extends to our Lab secretaries, Ms. Yoko Nakahara (for six years!), Ms. Rika Ikebe, and Ms. Michiyo Kai, for their indispensable assistance with administrative tasks.

Heartfelt thanks to everyone who provided mental and academic support during my PhD journey. This includes all my friends, colleagues and teachers who have been in our Lab, such as Associate Professor Toshiyuki Shimizu, and soccer teams Ohiru in Kyoto and Rising Sun in Atlanta. Also, thanks to Komeda Coffee Otsu Katata-Ten, which continuously provided a space where I could fully concentrate on my research in Shiga.

I would like to offer special thanks to Doctor Shun Takagi, an incredibly intelligent colleague and dear friend since high school. He has always been my closest mentor, providing me with a lot of invaluable advice. To Haruna, your love and encouragement have been my strength through the five challenging years, thanks as always. Lastly, I sincerely thank my family for their unwavering support throughout my long long student life.

Fumiyuki Kato, February 2024

References

[1] Meglena Kuneva. Roundtable on Online Data Collection, Targeting and Profiling, June 2009.
https://ec.europa.eu/commission/presscorner/detail/en/SPEECH_09_156, Accessed on 10.04.2023.

[2] Michelle Goddard. The eu general data protection regulation (gdpr): European regulation that has a global impact. *International Journal of Market Research*, 59(6):703–705, 2017.

[3] Office of the Attorney General. California Consumer Privacy Act (CCPA), 2018.
https://oag.ca.gov/privacy/ccpa, Accessed on 10.04.2023.

[4] Grace Dean. Amazon has been fined a record $887 million for violating data privacy rules in europe, July 2021.
https://www.businessinsider.com/amazon-eu-fine-data-privacy-gdpr-luxembourg-european-union-2021-7, Accessed on 10.04.2023.

[5] Chris Brook. Google fined $57m by data protection watchdog over gdpr violations, Dec 2022.
https://www.digitalguardian.com/blog/google-fined-57m-data-protection-watchdog-over-gdpr-violations, Accessed on 10.04.2023.

[6] IBM report. Cost of a data breach report 2023, 2023.
https://www.ibm.com/reports/data-breach, Accessed on 10.04.2023.

[7] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera

y Arcas. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629*, 2016.

[8] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[9] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pages 18–32. Springer, 2019.

[10] Li Huang, Andrew L Shea, Huining Qian, Aditya Masurkar, Hao Deng, and Dianbo Liu. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of biomedical informatics*, 99:103291, 2019.

[11] Dong Yang, Ziyue Xu, Wenqi Li, Andriy Myronenko, Holger R Roth, Stephanie Harmon, Sheng Xu, Baris Turkbey, Evrim Turkbey, Xiaosong Wang, et al. Federated semi-supervised learning for covid region segmentation in chest ct using multi-national data from china, italy, japan. *Medical image analysis*, 70:101992, 2021.

[12] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature medicine*, 27(10):1735–1743, 2021.

[13] Florian Hartmann and Google Research Peter Kairouz. Distributed differential privacy for federated learning, Mar 2023. `https://blog.research.google/2023/03/distributed-differential-privacy-for.html`, Accessed on 10.04.2023.

[14] Dzmitry Huba, John Nguyen, Kshitiz Malik, Ruiyu Zhu, Mike Rabbat, Ashkan Yousefpour, Carole-Jean Wu, Hongyuan Zhan, Pavel Ustinov, Har-

ish Srinivas, et al. Papaya: Practical, private, and scalable federated learning. *Proceedings of Machine Learning and Systems*, 4:814–832, 2022.

[15] LINE Corporation. Technical whitepaper: Differential privacy in line federated learning, Sep 2023. `https://d.line-scdn.net/stf/linecorp/en/csr/line-differential-privacy-whitepaper-ver1.0.pdf`, Accessed on 10.04.2023.

[16] Ewen Wang, Ajay Kannan, Yuefeng Liang, Boyi Chen, and Mosharaf Chowdhury. Flint: A platform for federated learning integration, 2023. To appear in MLSys 2023.

[17] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, et al. Federated evaluation and tuning for on-device personalization: System design & applications. *arXiv preprint arXiv:2102.08503*, 2021.

[18] Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Shanshan Han, Shantanu Sharma, Chaoyang He, Sharad Mehrotra, and Salman Avestimehr. Federated analytics: A survey, 2023.

[19] Google Research Daniel Ramage, Stefano Mazzocchi. Federated analytics: Collaborative data science without data collection, May 2020. `https://blog.research.google/2020/05/federated-analytics-collaborative-data.html`, Accessed on 10.04.2023.

[20] Ligeng Zhu and Song Han. Deep leakage from gradients. In *Federated learning*, pages 17–31. Springer, 2020.

[21] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. idlg: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[22] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947, 2020.

[23] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos Garcia Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-level label leakage from gradients in federated learning. *Proceedings on Privacy Enhancing Technologies*, 2022:227–244, 04 2022.

[24] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Olive: Oblivious federated learning on trusted execution environment against the risk of sparsification. *Proc. VLDB Endow.*, 16(10):2404–2417, aug 2023.

[25] Niv Haim, Gal Vardi, Gilad Yehudai, michal Irani, and Ohad Shamir. Reconstructing training data from trained neural networks. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[26] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[27] Umang Gupta, Dimitris Stripelis, Pradeep K. Lam, Paul Thompson, Jose Luis Ambite, and Greg Ver Steeg. Membership inference attacks on deep regression models for neuroimaging. In Mattias Heinrich, Qi Dou, Marleen de Bruijne, Jan Lellmann, Alexander Schläfer, and Floris Ernst, editors, *Proceedings of the Fourth Conference on Medical Imaging with Deep Learning*, volume 143 of *Proceedings of Machine Learning Research*, pages 228–251. PMLR, 07–09 Jul 2021.

[28] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. CCS '15, page 1322–1333, New York, NY, USA, 2015. Association for Computing Machinery.

[29] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[30] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data, Apr 2017.

https://blog.research.google/2017/04/federated-learning-collaborative.html, Accessed on 10.04.2023.

[31] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329*, 2019.

[32] F. Boenisch, A. Dziedzic, R. Schuster, A. Shahin Shamsabadi, I. Shumailov, and N. Papernot. When the curious abandon honesty: Federated learning is not private. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 175–199, Los Alamitos, CA, USA, jul 2023. IEEE Computer Society.

[33] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25*, pages 480–501. Springer, 2020.

[34] Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16070–16084. Curran Associates, Inc., 2020.

[35] Jacob Dumford and Walter Scheirer. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–9. IEEE, 2020.

[36] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

[37] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. Analyzing federated learning through an adversarial lens. In *International Conference on Machine Learning*, pages 634–643. PMLR, 2019.

[38] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900, 2021.

[39] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 947–964, 2021.

[40] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In *Data and Applications Security and Privacy XXXV: 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19–20, 2021, Proceedings 35*, pages 43–60. Springer, 2021.

[41] Zhaoxian Wu, Qing Ling, Tianyi Chen, and Georgios B Giannakis. Federated variance-reduced stochastic gradient descent with robustness to byzantine attacks. *IEEE Transactions on Signal Processing*, 68:4583–4596, 2020.

[42] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part II*, pages 1–12. Springer-Verlag, 2006.

[43] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kancha, Shraddha Sahay, and Parvez Ahammad. Linkedin's audience engagements api: A privacy preserving data analytics system at scale. *Journal of Privacy and Confidentiality*, 11, 12 2021.

[44] Alex Kulesza. Plume: Differential privacy at scale. Santa Clara, CA, September 2023. USENIX Association.

[45] Daniel Kifer, Solomon Messing, Aaron Roth, Abhradeep Thakurta, and Danfeng Zhang. Guidelines for implementing and auditing differentially private systems. *CoRR*, abs/2002.04049, 2020.

[46] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[47] Aref N Dajani, Amy D Lauger, Phyllis E Singer, Daniel Kifer, Jerome P Reiter, Ashwin Machanavajjhala, Simson L Garfinkel, Scot A Dahl, Matthew Graham, Vishesh Karwa, et al. The modernization of statistical disclosure limitation at the us census bureau. In *September 2017 meeting of the Census Scientific Advisory Committee*, 2017.

[48] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[49] Yu-Xiang Wang, Borja Balle, and Shiva Prasad Kasiviswanathan. Subsampled rényi differential privacy and analytical moments accountant. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1226–1235. PMLR, 2019.

[50] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th computer security foundations symposium (CSF)*, pages 263–275. IEEE, 2017.

[51] Ilya Mironov, Kunal Talwar, and Li Zhang. R\'enyi differential privacy of the sampled gaussian mechanism. *arXiv preprint arXiv:1908.10530*, 2019.

[52] Bargav Jayaraman and David Evans. Evaluating differentially private machine learning in practice. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1895–1912, Santa Clara, CA, August 2019. USENIX Association.

[53] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE symposium on security and privacy (SP)*, pages 656–672. IEEE, 2019.

[54] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[55] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[56] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 8(11):8836–8853, 2020.

[57] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Shuang Song, Kunal Talwar, and Abhradeep Thakurta. Encode, shuffle, analyze privacy revisited: Formalizations and empirical evaluation. *arXiv preprint arXiv:2001.03618*, 2020.

[58] Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *NIPS 2017 Workshop: Machine Learning on the Phone and other Consumer Devices*, 2017.

[59] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2468–2479. SIAM, 2019.

[60] Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. From bounded to unbounded: Privacy amplification via shuffling with dummies. In *2023 IEEE 36th Computer Security Foundations Symposium (CSF)*, pages 457–472. IEEE, 2023.

[61] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Network shuffling: Privacy amplification via random walks. In *Proceedings of the 2022 International Conference on Management of Data*, pages 773–787, 2022.

[62] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: what it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64. IEEE, 2015.

[63] Global Platform. Introduction to trusted execution environments, may 2018.
`https://globalplatform.org/resource-publication/introduction-to-trusted-execution-environments/`, Accessed on 10.06.2023.

[64] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1243–1255, 2017.

[65] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, volume 1, pages 448–457, 2001.

[66] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery.

[67] Mansouri Mohamad, Malek Onen, Wafa Ben Jaballah, and Mauro Contu. Sok: Secure aggregation based on cryptographic schemes for federated learning. In *Proceedings of Privacy Enhancing Technologies Symposium*, volume 1, 2023.

[68] Simone Bottoni, Giulio Zizzo, Stefano Braghin, and Alberto Trombetta. Verifiable federated learning. In *Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022)*, 2022.

[69] Wei Zheng, Ying Wu, Xiaoxue Wu, Chen Feng, Yulei Sui, Xiapu Luo, and Yajin Zhou. A survey of intel sgx and its applications. *Frontiers of Computer Science*, 15:1–15, 2021.

[70] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016(86):1–118, 2016.

[71] Lingchen Zhao, Jianlin Jiang, Bo Feng, Qian Wang, Chao Shen, and Qi Li. Sear: Secure and efficient aggregation for byzantine-robust federated learn-

ing. *IEEE Transactions on Dependable and Secure Computing*, 19(5):3329–3342, 2021.

[72] Chengliang Zhang, Junzhe Xia, Baichen Yang, Huancheng Puyang, Wei Wang, Ruichuan Chen, Istemi Ekin Akkus, Paarijaat Aditya, and Feng Yan. Citadel: Protecting data privacy and model confidentiality for collaborative learning. SoCC '21, page 546–561, New York, NY, USA, 2021. Association for Computing Machinery.

[73] Fan Mo, Hamed Haddadi, Kleomenis Katevas, Eduard Marin, Diego Perino, and Nicolas Kourtellis. Ppfl: Privacy-preserving federated learning with trusted execution environments. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '21, page 94–108, New York, NY, USA, 2021. Association for Computing Machinery.

[74] Yuhui Zhang, Zhiwei Wang, Jiangfeng Cao, Rui Hou, and Dan Meng. Shuffled: gradient-preserving federated learning using trusted execution environment. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, pages 161–168, 2021.

[75] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022.

[76] Tyler Hunt, Congzheng Song, Reza Shokri, Vitaly Shmatikov, and Emmett Witchel. Chiron: Privacy-preserving machine learning as a service. *arXiv preprint arXiv:1803.05961*, 2018.

[77] Fumiyuki Kato, Tsubasa Takahashi, Shun Takagi, Yang Cao, Seng Pei Liew, and Masatoshi Yoshikawa. Hdpview: Differentially private materialized view for exploring high dimensional relational data. *Proc. VLDB Endow.*, 15(9):1766–1778, may 2022.

[78] Fumiyuki Kato, Yang Cao, and Mastoshi Yoshikawa. Pct-tee: trajectory-based private contact tracing system with trusted execution environment. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(2):1–35, 2021.

[79] Fumiyuki Kato, Li Xiong, Shun Takagi, Yang Cao, and Masatoshi Yoshikawa. Uldp-fl: Federated learning with across silo user-level differential privacy. *arXiv preprint arXiv:2308.12210*, 2023.

[80] Yuhan Liu, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Michael Riley. Learning discrete distributions: user vs item-level privacy. *Advances in Neural Information Processing Systems*, 33:20965–20976, 2020.

[81] Daniel Levy, Ziteng Sun, Kareem Amin, Satyen Kale, Alex Kulesza, Mehryar Mohri, and Ananda Theertha Suresh. Learning with user-level privacy. *Advances in Neural Information Processing Systems*, 34:12466–12479, 2021.

[82] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private sql with bounded user contribution. *Proceedings on privacy enhancing technologies*, 2020(2):230–250, 2020.

[83] Kareem Amin, Alex Kulesza, Andres Munoz, and Sergei Vassilvtiskii. Bounding user contributions: A bias-variance trade-off in differential privacy. In *International Conference on Machine Learning*, pages 263–271. PMLR, 2019.

[84] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *2015 IEEE Symposium on Security and Privacy*, pages 640–656. IEEE, 2015.

[85] Jo Van Bulck, Nico Weichbrodt, Rüdiger Kapitza, Frank Piessens, and Raoul Strackx. Telling your secrets without page faults: Stealthy page table-based attacks on enclaved execution. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1041–1056, 2017.

[86] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. A survey of published attacks on intel sgx. *arXiv preprint arXiv:2006.13598*, 2020.

[87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 182–194, 1987.

[88] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[89] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *Proceedings of ICLR*, 2021.

[90] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep Gradient Compression: Reducing the communication bandwidth for distributed training. In *The International Conference on Learning Representations*, 2018.

[91] Atal Sahu, Aritra Dutta, Ahmed M Abdelmoniem, Trambak Banerjee, Marco Canini, and Panos Kalnis. Rethinking gradient sparsification as total error minimization. *Advances in Neural Information Processing Systems*, 34:8133–8146, 2021.

[92] Osama Shahid, Seyedamin Pouriyeh, Reza Meimandi Parizi, Quan Z. Sheng, Gautam Srivastava, and Liang Zhao. Communication efficiency in federated learning: Achievements and challenges. *ArXiv*, abs/2107.10996, 2021.

[93] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[94] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Fednas: Federated deep learning via neural architecture search. In *CVPR 2020 Workshop on Neural Architecture Search and Beyond for Representation Learning*, 2020.

[95] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. Back to the drawing board: A critical evaluation of poisoning attacks on production federated learning. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1354–1371. IEEE, 2022.

[96] Peter Kairouz, Ziyu Liu, and Thomas Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *International Conference on Machine Learning*, pages 5201–5212. PMLR, 2021.

[97] Irem Ergun, Hasin Us Sami, and Basak Guler. Sparsified secure aggregation for privacy-preserving federated learning. *arXiv preprint arXiv:2112.12872*, 2021.

[98] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2017.

[99] Meng Hao, Hongwei Li, Guowen Xu, Sen Liu, and Haomiao Yang. Towards efficient and privacy-preserving federated deep learning. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[100] Zachary Charles and Jakub Konečnỳ. Convergence and accuracy trade-offs in federated learning and meta-learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2575–2583. PMLR, 2021.

[101] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[102] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 965–978. IEEE, 2022.

[103] Rui Hu, Yanmin Gong, and Yuanxiong Guo. Federated learning with sparsified model perturbation: Improving accuracy under client-level differential privacy. *arXiv preprint arXiv:2202.07178*, 2022.

[104] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, et al. Fedml: A research library and benchmark for federated machine learning. *arXiv preprint arXiv:2007.13518*, 2020.

[105] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017*, 2018.

[106] Jean Ogier du Terrail, Samy-Safwan Ayed, Edwige Cyffers, Felix Grimberg, Chaoyang He, Regis Loeb, Paul Mangold, Tanguy Marchand, Othmane Marfoq, Erum Mushtaq, Boris Muzellec, Constantin Philippenko, Santiago Silva, Maria Teleńczuk, Shadi Albarqouni, Salman Avestimehr, Aurélien Bellet, Aymeric Dieuleveut, Martin Jaggi, Sai Praneeth Karimireddy, Marco Lorenzi, Giovanni Neglia, Marc Tommasi, and Mathieu Andreux. Flamby: Datasets and benchmarks for cross-silo federated learning in realistic healthcare settings. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5315–5334. Curran Associates, Inc., 2022.

[107] Ken Liu, Shengyuan Hu, Steven Z Wu, and Virginia Smith. On privacy and personalization in cross-silo federated learning. *Advances in Neural Information Processing Systems*, 35:5925–5940, 2022.

[108] Andrew Lowy, Ali Ghafelebashi, and Meisam Razaviyayn. Private nonconvex federated learning without a trusted server. In *International Conference on Artificial Intelligence and Statistics*, pages 5749–5786. PMLR, 2023.

[109] Brendan McMahan and Google Research Abhradeep Thakurta. Federated learning with formal differential privacy guarantees, Feb 2022.

`https://blog.research.google/2022/02/federated-learning-with-formal.html`, Accessed on 10.04.2023.

[110] Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30, 2009.

[111] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 51–60. IEEE, 2010.

[112] Borja Balle, Gilles Barthe, Marco Gaboardi, Justin Hsu, and Tetsuya Sato. Hypothesis testing interpretations and renyi differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 2496–2506. PMLR, 2020.

[113] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, June 2018.

[114] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data*, pages 155–170. ACM, 2016.

[115] Shun Takagi, Tsubasa Takahashi, Yang Cao, and Masatoshi Yoshikawa. P3gm: Private high-dimensional data release via privacy preserving phased generative model. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 169–180. IEEE, 2021.

[116] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. Privsyn: Differentially private data synthesis. In *USENIX Security Symposium*, 2021.

[117] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Pate-gan: generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018.

[118] Galen Andrew, Om Thakkar, H Brendan McMahan, and Swaroop Ramaswamy. Differentially private learning with adaptive clipping. *Advances in Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[119] Teng Wang, Xuefeng Zhang, Jingyu Feng, and Xinyu Yang. A comprehensive survey on local differential privacy toward data statistics and analysis. *Sensors*, 20(24):7030, Dec 2020.

[120] Ruixuan Liu, Yang Cao, Masatoshi Yoshikawa, and Hong Chen. Fedsel: Federated sgd under local differential privacy with top-k dimension selection. In *International Conference on Database Systems for Advanced Applications*, pages 485–501. Springer, 2020.

[121] Lichao Sun, Jianwei Qian, and Xun Chen. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1571–1578. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[122] Ruixuan Liu, Yang Cao, Hong Chen, Ruoyang Guo, and Masatoshi Yoshikawa. Flame: Differentially private federated learning in the shuffle model. In *AAAI*, 2021.

[123] Antonious Girgis, Deepesh Data, Suhas Diggavi, Peter Kairouz, and Ananda Theertha Suresh. Shuffled model of differential privacy in federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2521–2529. PMLR, 2021.

[124] Borja Balle, James Bell, Adrià Gascón, and Kobbi Nissim. The privacy blanket of the shuffle model. In *Annual International Cryptology Conference*, pages 638–667. Springer, 2019.

[125] Wei-Ning Chen, Christopher A Choquette Choo, Peter Kairouz, and Ananda Theertha Suresh. The fundamental price of secure aggregation in differentially private federated learning. In *International Conference on Machine Learning*, pages 3056–3089. PMLR, 2022.

[126] Zibo Wang, Yifei Zhu, Dan Wang, and Zhu Han. Fedfpm: A unified federated analytics framework for collaborative frequent pattern mining. In

IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, pages 61–70, 2022.

[127] Wennan Zhu, Peter Kairouz, Brendan McMahan, Haicheng Sun, and Wei Li. Federated heavy hitters discovery with differential privacy. In *International Conference on Artificial Intelligence and Statistics*, pages 3837–3847. PMLR, 2020.

[128] Sandro Pinto and Nuno Santos. Demystifying arm trustzone: A comprehensive survey. *ACM computing surveys (CSUR)*, 51(6):1–36, 2019.

[129] David Kaplan, Jeremy Powell, and Tom Woller. Amd memory encryption. *White paper*, page 13, 2016.

[130] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 857–874, 2016.

[131] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. Keystone: An open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems*, pages 1–16, 2020.

[132] Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, and Frank Mckeen. Intel software guard extensions: Epid provisioning and attestation services. *White Paper*, 1(1-10):119, 2016.

[133] Kajetan Maliszewski, Jorge-Arnulfo Quiané-Ruiz, Jonas Traub, and Volker Markl. What is the price for joining securely? benchmarking equi-joins in trusted execution environments. *Proceedings of the VLDB Endowment*, 15(3):659–672, 2021.

[134] Meysam Taassori, Ali Shafiee, and Rajeev Balasubramonian. Vault: Reducing paging overheads in sgx with efficient integrity verification structures. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 665–678, 2018.

[135] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, et al. Decentralized privacy-preserving proximity tracing. *arXiv preprint arXiv:2005.12273*, 2020.

[136] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure:sgx cache attacks are practical. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[137] Sangho Lee, Ming-Wei Shih, Prasun Gera, Taesoo Kim, Hyesoon Kim, and Marcus Peinado. Inferring fine-grained control flow inside sgx enclaves with branch shadowing. In *USENIX Security Symposium*, volume 19, pages 16–18, 2017.

[138] Pau-Chen Cheng, Kevin Eykholt, Zhongshu Gu, Hani Jamjoom, K. R. Jayaram, Enriquillo Valdez, and Ashish Verma. Separation of powers in federated learning (poster paper). In *Proceedings of the First Workshop on Systems Challenges in Reliable and Secure Federated Learning*, ResilientFL '21, page 16–18, New York, NY, USA, 2021. Association for Computing Machinery.

[139] A. Mondal, Y. More, R. Rooparaghunath, and D. Gupta. Poster: Flatee: Federated learning across trusted execution environments. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 707–709, Los Alamitos, CA, USA, sep 2021. IEEE Computer Society.

[140] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 619–636, 2016.

[141] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[142] H Brendan McMahan, Galen Andrew, Ulfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. A general approach to

adding differential privacy to iterative training procedures. *arXiv preprint arXiv:1812.06210*, 2018.

[143] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[144] Andrew Lowy and Meisam Razaviyayn. Private federated learning without a trusted server: Optimal algorithms for convex losses. In *The Eleventh International Conference on Learning Representations*, 2023.

[145] Gautam Kamath. Cs 860 : Algorithms for private data analysis fall 2020 lecture 5 — approximate differential privacy. `http://www.gautamkamath.com/CS860notes/lec5.pdf`, 2020. [Online; accessed 23-June-2023].

[146] Naman Agarwal, Peter Kairouz, and Ziyu Liu. The skellam mechanism for differentially private federated learning. *Advances in Neural Information Processing Systems*, 34:5052–5064, 2021.

[147] James Henry Bell, Kallista A Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. Secure single-server aggregation with (poly) logarithmic overhead. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1253–1269, 2020.

[148] Albert Cheu, Adam Smith, Jonathan Ullman, David Zeber, and Maxim Zhilyaev. Distributed differential privacy via shuffling. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38*, pages 375–403. Springer, 2019.

[149] Jinhyun So, Ramy E Ali, Başak Güler, Jiantao Jiao, and A Salman Avestimehr. Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9864–9873, 2023.

[150] Dinusha Vatsalan, Ziad Sehili, Peter Christen, and Erhard Rahm. Privacy-preserving record linkage for big data: Current approaches and research challenges. *Handbook of big data technologies*, pages 851–895, 2017.

[151] Kamalika Chaudhuri, Jacob Imola, and Ashwin Machanavajjhala. *Capacity Bounded Differential Privacy*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[152] Alessandro Epasto, Mohammad Mahdian, Jieming Mao, Vahab Mirrokni, and Lijie Ren. Smoothly bounding user contributions in differential privacy. *Advances in Neural Information Processing Systems*, 33:13999–14010, 2020.

[153] Xinwei Zhang, Xiangyi Chen, Mingyi Hong, Zhiwei Steven Wu, and Jinfeng Yi. Understanding clipping for federated learning: Convergence and client-level differential privacy. In *International Conference on Machine Learning, ICML 2022*, 2022.

[154] Sashank Reddi, Zachary Burr Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Brendan McMahan, editors. *Adaptive Federated Optimization*, 2021.

[155] Andreea B. Alexandru and George J. Pappas. Private weighted sum aggregation. *IEEE Transactions on Control of Network Systems*, 9(1):219–230, 2022.

[156] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy: 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007. Proceedings 12*, pages 416–430. Springer, 2007.

[157] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[158] Kaggle. Credit card fraud detection dataset. `https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud`, 2018. Accessed: 2023-08-03.

[159] Zhaoxiong Yang, Shuihai Hu, and Kai Chen. Fpga-based hardware accelerator of homomorphic encryption for efficient federated learning. *arXiv preprint arXiv:2007.10560*, 2020.

[160] Franziska Boenisch, Christopher Mühl, Adam Dziedzic, Roy Rinberg, and Nicolas Papernot. Have it your way: Individualized privacy assignment for dp-sgd. *arXiv preprint arXiv:2303.17046*, 2023.

[161] Shiwei Lu, Ruihu Li, Wenbin Liu, Chaofeng Guan, and Xiaopeng Yang. Top-k sparsification with secure aggregation for privacy-preserving federated learning. *Computers & Security*, 124:102993, 2023.

[162] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path oram: An extremely simple oblivious ram protocol. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 299–310, New York, NY, USA, 2013. Association for Computing Machinery.

[163] Sajin Sasy, Sergey Gorbunov, and Christopher W. Fletcher. Zerotrace: Oblivious memory primitives from intel SGX. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[164] Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *NSDI*, volume 17, pages 283–298, 2017.

[165] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

[166] Chong Fu, Xuhong Zhang, Shouling Ji, Jinyin Chen, Jingzheng Wu, Shanqing Guo, Jun Zhou, Alex X Liu, and Ting Wang. Label inference attacks against vertical federated learning. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1397–1414, 2022.

[167] Kenneth E Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 307–314, 1968.

[168] Donglei Wu, Xiangyu Zou, Shuyu Zhang, Haoyu Jin, Wen Xia, and Binxing Fang. Smartidx: Reducing communication cost in federated learning by exploiting the cnns structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4254–4262, 2022.

[169] TH Hubert Chan, Kai-Min Chung, Bruce M Maggs, and Elaine Shi. Foundations of differentially oblivious algorithms. In *Proceedings of the Thirtieth*

Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2448–2467. SIAM, 2019.

[170] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR, 26–28 Aug 2020.

[171] Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. *arXiv preprint arXiv:2112.02918*, 2021.

[172] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE INFOCOM 2019-IEEE conference on computer communications*, pages 2512–2520. IEEE, 2019.

[173] Shaohuai Shi, Kaiyong Zhao, Qiang Wang, Zhenheng Tang, and Xiaowen Chu. A convergence analysis of distributed sgd with communication-efficient gradient sparsification. In *IJCAI*, pages 3411–3417, 2019.

[174] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618, 2017.

[175] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[176] Joshua Allen, Bolin Ding, Janardhan Kulkarni, Harsha Nori, Olga Ohrimenko, and Sergey Yekhanin. An algorithmic framework for differentially private data analysis on trusted processors. *Advances in Neural Information Processing Systems*, 32, 2019.

[177] Sahar Mazloom and S. Dov Gordon. Secure computation with differentially private access patterns. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 490–507, New York, NY, USA, 2018. Association for Computing Machinery.

[178] Shumo Chu, Danyang Zhuo, Elaine Shi, and TH Hubert Chan. Differentially oblivious database joins: Overcoming the worst-case curse of fully oblivious algorithms. *Cryptology ePrint Archive*, 2021.

[179] Lianke Qin, Rajesh Jayaram, Elaine Shi, Zhao Song, Danyang Zhuo, and Shumo Chu. Adore: Differentially oblivious relational database operators. *Proc. VLDB Endow.*, 16(4):842–855, dec 2022.

[180] Benjamin M Case, James Honaker, and Mahnush Movahedi. The privacy-preserving padding problem: Non-negative mechanisms for conservative answers with differential privacy. *arXiv preprint arXiv:2110.08177*, 2021.

[181] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. Sgx-bomb: Locking down the processor via rowhammer attack. In *Proceedings of the 2nd Workshop on System Software for Trusted Execution*, pages 1–6, 2017.

[182] Guoxing Chen, Sanchuan Chen, Yuan Xiao, Yinqian Zhang, Zhiqiang Lin, and Ten H Lai. Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 142–157. IEEE, 2019.

[183] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution. In *Proceedings fo the 27th USENIX Security Symposium*. USENIX Association, 2018.

[184] Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based fault injection attacks against intel sgx. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1466–1482. IEEE, 2020.

[185] Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D Garcia, and Frank Piessens. A tale of two worlds: Assessing the vulnerabil-

ity of enclave shielding runtimes. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1741–1758, 2019.

[186] Anda Cheng, Peisong Wang, Xi Sheryl Zhang, and Jian Cheng. Differentially private federated learning with local regularization and sparsification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10122–10131, 2022.

[187] Lingxiao Wang, Bargav Jayaraman, David Evans, and Quanquan Gu. Efficient privacy-preserving stochastic nonconvex optimization. *arXiv preprint arXiv:1910.13659*, 2019.

[188] Lili Su and Jiaming Xu. Securing distributed gradient descent in high dimensional statistical learning. *Proc. ACM Meas. Anal. Comput. Syst.*, 3(1), mar 2019.

[189] Abhishek Bhowmick, John Duchi, Julien Freudiger, Gaurav Kapoor, and Ryan Rogers. Protection against reconstruction and its applications in private federated learning. *arXiv preprint arXiv:1812.00984*, 2018.

[190] Ashay Rane, Calvin Lin, and Mohit Tiwari. Raccoon: Closing digital side-channels through obfuscated execution. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 431–446, 2015.

[191] Rohit Sinha, Sriram Rajamani, and Sanjit A Seshia. A compiler and verifier for page access oblivious computation. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 649–660, 2017.

[192] Sajin Sasy, Aaron Johnson, and Ian Goldberg. Fast fully oblivious compaction and shuffling. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2565–2579, 2022.

[193] Sajin Sasy, Aaron Johnson, and Ian Goldberg. Waks-on/waks-off: Fast oblivious offline/online shuffling and sorting with waksman networks. Cryptology ePrint Archive, Paper 2023/1236, 2023. `https://eprint.iacr.org/2023/1236`.

[194] Do Le Quoc and Christof Fetzer. Secfl: Confidential federated learning using tees. *arXiv preprint arXiv:2110.00981*, 2021.

[195] Alexandre Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 211–222, 2003.

[196] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[197] Apple Differential Privacy Team. Learning with Privacy at Scale, Dec 2017.
`https://machinelearning.apple.com/research/learning-with-privacy-at-scale`, Accessed on 10.04.2023.

[198] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. *Advances in Neural Information Processing Systems*, 30, 2017.

[199] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[200] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *Journal of Machine Learning Research*, 17(17):1–51, 2016.

[201] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally differentially private protocols for frequency estimation. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 729–745, 2017.

[202] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. Discrete distribution estimation under local privacy. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2436–2444, New York, New York, USA, 20–22 Jun 2016. PMLR.

[203] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. Answering multi-dimensional analytical queries under local differential privacy. In *Proceedings of the 2019 International Conference on Management of Data*, pages 159–176, 2019.

[204] Tianhao Wang, Ninghui Li, and Somesh Jha. Locally differentially private frequent itemset mining. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 127–143. IEEE, 2018.

[205] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery.

[206] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques. In *Public Key Cryptography–PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004. Proceedings 7*, pages 425–438. Springer, 2004.

[207] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[208] Drazen Prelec. A bayesian truth serum for subjective data. *science*, 306(5695):462–466, 2004.

[209] Cuong T. Do, Nguyen H. Tran, Choongseon Hong, Charles A. Kamhoua, Kevin A. Kwiat, Erik Blasch, Shaolei Ren, Niki Pissinou, and Sundaraja Sitharama Iyengar. Game theory for cyber security and privacy. *ACM Comput. Surv.*, 50(2), may 2017.

[210] Dalia Attia Waguih and Laure Berti-Équille. Truth discovery algorithms: An experimental evaluation. *CoRR*, abs/1409.6428, 2014.

[211] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78(110), 1998.

[212] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[213] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, pages 174–187, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[214] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, 1990.

[215] Gonzalo Munilla Garrido, Johannes Sedlmeir, and Matthias Babel. Towards verifiable differentially-private polling. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–11, 2022.

[216] Danielle Movsowitz Davidow, Yacov Manevich, and Eran Toch. Privacy-preserving payment system with verifiable local differential privacy. *Cryptology ePrint Archive*, 2023.

[217] Xiaoli Zhang, Fengting Li, Zeyu Zhang, Qi Li, Cong Wang, and Jianping Wu. Enabling execution assurance of federated learning at untrusted participants. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 1877–1886, 2020.

[218] Zhibo Xing, Zijian Zhang, Meng Li, Jiamou Liu, Liehuang Zhu, Giovanni Russello, and Muhammad Rizwan Asghar. Zero-knowledge proof-based practical federated learning on blockchain. *arXiv preprint arXiv:2304.05590*, 2023.

[219] Anran Li, Lan Zhang, Junhao Wang, Juntao Tan, Feng Han, Yaxuan Qin, Nikolaos M Freris, and Xiang-Yang Li. Efficient federated-learning model debugging. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 372–383. IEEE, 2021.

[220] Jingwen Zhang, Yuezhou Wu, and Rong Pan. Incentive mechanism for horizontal federated learning based on reputation and reverse auction. In *Proceedings of the Web Conference 2021*, pages 947–956, 2021.

[221] Weishan Zhang, Qinghua Lu, Qiuyu Yu, Zhaotong Li, Yue Liu, Sin Kit Lo, Shiping Chen, Xiwei Xu, and Liming Zhu. Blockchain-based federated learning for device failure detection in industrial iot. *IEEE Internet of Things Journal*, 8(7):5926–5937, 2020.

[222] Sin Kit Lo, Yue Liu, Qinghua Lu, Chen Wang, Xiwei Xu, Hye-Young Paik, and Liming Zhu. Blockchain-based trustworthy federated learning architecture. *arXiv preprint arXiv:2108.06912*, 2021.

[223] Shuyuan Zheng, Yang Cao, Masatoshi Yoshikawa, Huizhong Li, and Qiang Yan. Fl-market: Trading private models in federated learning. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1525–1534. IEEE, 2022.

[224] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[225] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. Pmf: A privacy-preserving human mobility prediction framework via federated learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–21, 2020.

[226] Tianlong Yu, Tian Li, Yuqiong Sun, Susanta Nanda, Virginia Smith, Vyas Sekar, and Srinivasan Seshan. Learning context-aware policies from multiple smart homes via federated multi-task learning. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 104–115. IEEE, 2020.

[227] Konstantin Sozinov, Vladimir Vlassov, and Sarunas Girdzijauskas. Human activity recognition using federated learning. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 1103–1111. IEEE, 2018.

[228] Stefano Savazzi, Monica Nicoli, Mehdi Bennis, Sanaz Kianoush, and Luca Barbieri. Opportunities of federated learning in connected, cooperative, and automated industrial systems. *IEEE Communications Magazine*, 59(2):16–21, 2021.

[229] Tianchi Deng, Yingguang Li, Xu Liu, and Lihui Wang. Federated learning-based collaborative manufacturing for complex parts. *Journal of Intelligent Manufacturing*, 34(7):3025–3038, 2023.

[230] Xu Han, Haoran Yu, and Haisong Gu. Visual inspection with federated learning. In *Image Analysis and Recognition: 16th International Conference, ICIAR 2019, Waterloo, ON, Canada, August 27–29, 2019, Proceedings, Part II 16*, pages 52–64. Springer, 2019.

[231] Anh Nguyen, Tuong Do, Minh Tran, Binh X. Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, and Quang D. Tran. Deep federated learning for autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1824–1830, 2022.

[232] Stephen R Pfohl, Andrew M Dai, and Katherine Heller. Federated and differentially private learning for electronic health records. *arXiv preprint arXiv:1911.05861*, 2019.

[233] Mohammed Adnan, Shivam Kalra, Jesse C Cresswell, Graham W Taylor, and Hamid R Tizhoosh. Federated learning and differential privacy for medical image analysis. *Scientific reports*, 12(1):1953, 2022.

[234] Satyabrata Aich, Nday Kabulo Sinai, Saurabh Kumar, Mohammed Ali, Yu Ran Choi, Moon-IL Joo, and Hee-Cheol Kim. Protecting personal healthcare record using blockchain & federated learning technologies. In *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pages 109–112. IEEE, 2022.

[235] Hao-Ping Lee, Yu-Ju Yang, Thomas Serban von Davier, Jodi Forlizzi, and Sauvik Das. Deepfakes, phrenology, surveillance, and more! a taxonomy of ai privacy risks. *arXiv preprint arXiv:2310.07879*, 2023.

[236] Du Su, Hieu Tri Huynh, Ziao Chen, Yi Lu, and Wenmiao Lu. Re-identification attack to privacy-preserving data analysis with noisy sample-

mean. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1045–1053, 2020.

[237] Zhen Wang, Xiang Yue, Soheil Moosavinasab, Yungui Huang, Simon Lin, and Huan Sun. Surfcon: Synonym discovery on privacy-aware clinical data. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1578–1586, 2019.

[238] Chang Ge, Xi He, Ihab F Ilyas, and Ashwin Machanavajjhala. Apex: Accuracy-aware differentially private data exploration. In *Proceedings of the 2019 International Conference on Management of Data*, pages 177–194, 2019.

[239] Royce Wilson, Celia Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. Differentially private sql with bounded user contribution. *Proceedings on Privacy Enhancing Technologies*, 2020:230–250, 04 2020.

[240] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. Linkedin's audience engagements api: A privacy preserving data analytics system at scale. *arXiv preprint arXiv:2002.05839*, 2020.

[241] Noah Johnson, Joseph P Near, Joseph M Hellerstein, and Dawn Song. Chorus: Differential privacy via query rewriting. *arXiv preprint arXiv:1809.07750*, 2018.

[242] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):25, 2017.

[243] Xiaojian Zhang, Rui Chen, Jianliang Xu, Xiaofeng Meng, and Yingtao Xie. Towards accurate histogram publication under differential privacy. In Mohammed Javeed Zaki, Zoran Obradovic, Pang-Ning Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy, editors, *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 587–595. SIAM, 2014.

[244] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Differentially private grids for geospatial data. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 757–768. IEEE, 2013.

[245] Yonghui Xiao, Li Xiong, Liyue Fan, and Slawomir Goryczka. Dpcube: Differentially private histogram release through multidimensional partitioning. *arXiv preprint arXiv:1202.5358*, 2012.

[246] Grigory Yaroslavtsev, Graham Cormode, Cecilia M Procopiuc, and Divesh Srivastava. Accurate and efficient private release of datacubes and contingency tables. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 745–756. IEEE, 2013.

[247] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment*, 12(11):1371–1384, 2019.

[248] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. A data- and workload-aware algorithm for range queries under differential privacy. *Proc. VLDB Endow.*, 7(5):341–352, January 2014.

[249] Chao Li, Gerome Miklau, Michael Hay, Andrew McGregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24(6):757–781, 2015.

[250] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446, 2014.

[251] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138. ACM, 2015.

[252] Frederik Harder, Kamil Adamczewski, and Mijung Park. Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-

preserving data generation. In *International Conference on Artificial Intelligence and Statistics*, pages 1819–1827. PMLR, 2021.

[253] Ju Fan, Junyou Chen, Tongyu Liu, Yuwei Shen, Guoliang Li, and Xiaoyong Du. Relational data synthesis using generative adversarial networks: A design space exploration. *Proc. VLDB Endow.*, 13(12):1962–1975, July 2020.

[254] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F Ilyas. Kamino: Constraint-aware differentially private data synthesis. *arXiv preprint arXiv:2012.15713*, 2020.

[255] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. Differentially private spatial decompositions. In *2012 IEEE 28th International Conference on Data Engineering*, pages 20–31. IEEE, 2012.

[256] Chugui Xu, Ju Ren, Yaoxue Zhang, Zhan Qin, and Kui Ren. Dppro: Differentially private high-dimensional data release via random projection. *IEEE Transactions on Information Forensics and Security*, 12(12):3081–3093, 2017.

[257] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, 2018.

[258] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

[259] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26, 1984.

[260] Zhigao Zheng, Tao Wang, Jinming Wen, Shahid Mumtaz, Ali Kashif Bashir, and Sajjad Hussain Chauhdary. Differentially private high-dimensional data publication in internet of things. *IEEE Internet of Things Journal*, 7(4):2640–2650, 2019.

[261] Egon Balas and Manfred W Padberg. Set partitioning: A survey. *SIAM review*, 18(4):710–760, 1976.

[262] C Jason Wang, Chun Y Ng, and Robert H Brook. Response to covid-19 in taiwan: big data analytics, new technology, and proactive testing. *Jama*, 323(14):1341–1342, 2020.

[263] Chuan Qin, Luoqi Zhou, Ziwei Hu, Shuoqi Zhang, Sheng Yang, Yu Tao, Cuihong Xie, Ke Ma, Ke Shang, Wei Wang, and Dai-Shi Tian. Dysregulation of Immune Response in Patients With Coronavirus 2019 (COVID-19) in Wuhan, China. *Clinical Infectious Diseases*, 71(15):762–768, 03 2020.

[264] Marcel Salathé, Christian L Althaus, Richard Neher, Silvia Stringhini, Emma Hodcroft, Jacques Fellay, Marcel Zwahlen, Gabriela Senti, Manuel Battegay, Annelies Wilder-Smith, et al. Covid-19 epidemic in switzerland: on the importance of testing, contact tracing and isolation. *Swiss medical weekly*, 150(11-12):w20225, 2020.

[265] Hyunghoon Cho, Daphne Ippolito, and Yun William Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. *arXiv preprint arXiv:2003.11511*, 2020.

[266] Luca Ferretti, Chris Wymant, Michelle Kendall, Lele Zhao, Anel Nurtay, Lucie Abeler-Dörner, Michael Parker, David Bonsall, and Christophe Fraser. Quantifying sars-cov-2 transmission suggests epidemic control with digital contact tracing. *Science*, 368(6491), 2020.

[267] L. Reichert, Samuel Brack, and B. Scheuermann. Privacy-preserving contact tracing of covid-19 patients. *Presented at the Poster Session at the 41st IEEE Symposium on Security and Privacy.*, 2020.

[268] Qiang Tang. Privacy-preserving contact tracing: current solutions and open questions. *arXiv preprint arXiv:2004.06818*, 2020.

[269] Samuel Brack, Leonie Reichert, and Björn Scheuermann. Decentralized contact tracing using a dht and blind signatures. *IACR Cryptol. ePrint Arch.*, 2020:398, 2020.

[270] HUMAN RIGHTS WATCH. Mobile location data and covid-19: Q&a, 2020. `https://www.hrw.org/news/2020/05/13/mobile-location-data-and-covid-19-qa`.

[271] Rachael Thorneloe, Tracy Epton, Wendy Fynn, Michael Daly, Natalia Stanulewicz, Angelos Kassianos, Gillian W Shorter, Sarah-Jane Moll, Miglena Campbell, Samantha Sodergren, et al. Scoping review of mobile phone app uptake and engagement to inform digital contact tracing tools for covid-19. 2020.

[272] Christian S Jensen, Hua Lu, and Man Lung Yiu. Location privacy techniques in client-server architectures. In *Privacy in location-based applications*, pages 31–58. Springer, 2009.

[273] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *arXiv preprint arXiv:2004.13293*, 2020.

[274] Ronald L Rivest, Jon Callas, Ran Canetti, Kevin Esvelt, Daniel Kahn Gillmor, Yael Tauman Kalai, Anna Lysyanskaya, Adam Norige, Ramesh Raskar, Adi Shamir, et al. The pact protocol specification. *Private Automated Contact Tracing Team, MIT, Cambridge, MA, USA, Tech. Rep. 0.1*, 2020.

[275] Johannes K Becker, David Li, and David Starobinski. Tracking anonymized bluetooth devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[276] Yaron Gvili. Security analysis of the covid-19 contact tracing specifications by apple inc. and google inc. *IACR Cryptol. ePrint Arch.*, 2020:428, 2020.

[277] TCNCoalition. Tcn, 2020. `https://github.com/TCNCoalition/TCN`.

[278] Centers for Disease Control and Prevention. Cdc updates covid-19 transmission webpage to clarify information about types of spread, 2020. `https://www.cdc.gov/media/releases/2020/s0522-cdc-updates-covid-transmission.html`.

[279] Neeltje Van Doremalen, Trenton Bushmaker, Dylan H Morris, Myndi G Holbrook, Amandine Gamble, Brandi N Williamson, Azaibi Tamin, Jennifer L Harcourt, Natalie J Thornburg, Susan I Gerber, et al. Aerosol and surface stability of sars-cov-2 as compared with sars-cov-1. *New England journal of medicine*, 382(16):1564–1567, 2020.

[280] Chaojun Xie, Hongjun Zhao, Kuibiao Li, Zhoubin Zhang, Xiaoxiao Lu, Huide Peng, Dahu Wang, Jin Chen, Xiao Zhang, Di Wu, et al. The evidence of indirect transmission of sars-cov-2 reported in guangzhou, china. *BMC public health*, 20(1):1–9, 2020.

[281] Brittany Falkers. How covid-19 cases are evolving, along with our understanding of the virus, 2020. `https://www.kgw.com/article/news/health/coronavirus/multnomah-co-top-health-official-talks-about-our-evolving-understanding-of-the-novel-coronavirus/283-f2deae47-ff37-4699-9b4b-12e0c522f03c`.

[282] The Japan Times. New who guidance calls for more evidence on airborne coronavirus transmission, 2020. `https://www.japantimes.co.jp/news/2020/07/10/world/science-health-world/who-covid19-airborne-transmission/`.

[283] Konstantin D Pandl, Scott Thiebes, Manuel Schmidt-Kraepelin, and Ali Sunyaev. How detection ranges and usage stops impact digital contact tracing effectiveness for covid-19. *Scientific reports*, 11(1):1–11, 2021.

[284] Mirco Nanni, Gennady Andrienko, Chiara Boldrini, Francesco Bonchi, Ciro Cattuto, Francesca Chiaromonte, Giovanni Comandé, Marco Conti, Mark Coté, Frank Dignum, et al. Give more data, awareness and control to individual citizens, and they will help covid-19 containment. *arXiv preprint arXiv:2004.05222*, 2020.

[285] Li Xiong, Cyrus Shahabi, Yanan Da, Ritesh Ahuja, Vicki Hertzberg, Lance Waller, Xiaoqian Jiang, and Amy Franklin. React: Real-time contact tracing and risk monitoring using privacy-enhanced mobile tracking. *SIGSPATIAL Special*, 12(2):3–14, October 2020.

[286] Roba Abbas and Katina Michael. Covid-19 contact trace app deployments: Learnings from australia and singapore. *IEEE Consumer Electronics Magazine*, 9(5):65–70, 2020.

[287] Peter Rindal and Mike Rosulek. Malicious-secure private set intersection via dual execution. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1229–1242, 2017.

[288] Sandeep Tamrakar, Jian Liu, Andrew Paverd, Jan-Erik Ekberg, Benny Pinkas, and N Asokan. The circle game: Scalable private membership test using trusted hardware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 31–44, 2017.

[289] Arvind Narayanan, Narendran Thiagarajan, Mugdha Lakhani, Michael Hamburg, Dan Boneh, et al. Location privacy via private proximity testing. In *NDSS*, volume 11, 2011.

[290] Huanchen Zhang, Hyeontaek Lim, Viktor Leis, David G Andersen, Michael Kaminsky, Kimberly Keeton, and Andrew Pavlo. Surf: Practical range query filtering with fast succinct tries. In *Proceedings of the 2018 International Conference on Management of Data*, pages 323–336, 2018.

[291] Hongliang Tian, Qiong Zhang, Shoumeng Yan, Alex Rudnitsky, Liron Shacham, Ron Yariv, and Noam Milshten. Switchless calls made practical in intel sgx. In *Proceedings of the 3rd Workshop on System Software for Trusted Execution*, pages 22–27, 2018.

[292] Kevin Lahey. Monitoring intel sgx enclaves, 2020. `https://fortanix.com/blog/2020/02/monitoring-intel-sgx-enclaves/`.

[293] Shay Gueron. Memory encryption for general-purpose processors. *IEEE Security & Privacy*, 14(6):54–62, 2016.

[294] Anders T Gjerdrum, Robert Pettersen, Håvard D Johansen, and Dag Johansen. Performance of trusted computing in cloud infrastructures with intel sgx. In *CLOSER*, pages 668–675, 2017.

[295] Can Kockan, Kaiyuan Zhu, Natnatee Dokmai, Nikolai Karpov, M Oguzhan Kulekci, David P Woodruff, and S Cenk Sahinalp. Sketching algorithms for genomic data analysis and querying in a secure enclave. *Nature Methods*, 17(3):295–301, 2020.

[296] Emiliano De Cristofaro, Jihye Kim, and Gene Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 213–231. Springer, 2010.

[297] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS*, 2012.

[298] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on OT extension. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 797–812, San Diego, CA, August 2014. USENIX Association.

[299] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on ot extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[300] Payman Mohassel and Matthew Franklin. Efficiency tradeoffs for malicious two-party computation. In *International Workshop on Public Key Cryptography*, pages 458–473. Springer, 2006.

[301] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237, 2018.

[302] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. Pir-psi: scaling private contact discovery. *Proceedings on Privacy Enhancing Technologies*, 2018(4):159–178, 2018.

[303] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[304] Pramod Subramanyan, Rohit Sinha, Ilia Lebedev, Srinivas Devadas, and Sanjit A Seshia. A formal foundation for secure remote execution of enclaves. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450, 2017.

[305] Centers for Disease Control and Prevention. Public health guidance for community-related exposure, 2021. `https://www.cdc.gov/coronavirus/2019-ncov/php/public-health-recommendations.html`.

[306] Mark Zastrow. South korea is reporting intimate details of covid-19 cases: has it helped?, 2020. `https://www.nature.com/articles/d41586-020-00740-y`.

[307] Muge Cevik, Julia Marcus, Caroline Buckee, and Tara Smith. Sars-cov-2 transmission dynamics should inform policy. *Available at SSRN 3692807*, 2020.

[308] Serge Vaudenay. Analysis of dp3t. *IACR Cryptol. ePrint Arch.*, 2020:399, 2020.

[309] Gennaro Avitabile, Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. Towards defeating mass surveillance and sars-cov-2: The pronto-c2 fully decentralized automatic contact tracing system. *IACR Cryptol. ePrint Arch.*, 2020:493, 2020.

[310] Mohsen Ali, Tamer ElBatt, and Moustafa Youssef. Senseio: Realistic ubiquitous indoor outdoor detection system using smartphones. *IEEE Sensors Journal*, 18(9):3684–3693, 2018.

[311] Xingyu Huang, Yong Li, Yue Wang, Xinlei Chen, Yu Xiao, and Lin Zhang. Cts: A cellular-based trajectory tracking system with gps-level accuracy. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(4):1–29, 2018.

[312] Tarun Kulshrestha, Divya Saxena, Rajdeep Niyogi, Vaskar Raychoudhury, and Manoj Misra. Smartits: Smartphone-based identification and tracking using seamless indoor-outdoor localization. *Journal of Network and Computer Applications*, 98:97–113, 2017.

[313] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Secure and efficient trajectory-based contact tracing using trusted hardware. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4016–4025. IEEE, 2020.

[314] Xuefeng Guan, Cheng Bo, Zhenqiang Li, and Yaojin Yu. St-hash: An efficient spatiotemporal index for massive trajectory data in a nosql database. In *2017 25th International Conference on Geoinformatics*, pages 1–7. IEEE, 2017.

[315] Joe Schwartz(Microsoft). Bing maps tile system, 2018. `https://docs.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system`.

[316] Huibo Wang, Pei Wang, Yu Ding, Mingshen Sun, Yiming Jing, Ran Duan, Long Li, Yulong Zhang, Tao Wei, and Zhiqiang Lin. Towards memory safe enclave programming with rust-sgx. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2333–2350, 2019.

[317] Luca Pappalardo, Filippo Simini, Salvatore Rinzivillo, Dino Pedreschi, Fosca Giannotti, and Albert-László Barabási. Returners and explorers dichotomy in human mobility. *Nature communications*, 6(1):1–8, 2015.

[318] Yoshihide Sekimoto, Ryosuke Shibasaki, Hiroshi Kanasugi, Tomotaka Usui, and Yasunobu Shimazaki. Pflow: Reconstructing people flow recycling large-scale social survey data. *IEEE Pervasive Computing*, 10(4):27–35, 2011.

[319] Jack Doerner and Abhi Shelat. Scaling oram for secure computation. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 523–535, 2017.

[320] Raad Bahmani, Manuel Barbosa, Ferdinand Brasser, Bernardo Portela, Ahmad-Reza Sadeghi, Guillaume Scerri, and Bogdan Warinschi. Secure multiparty computation from sgx. In *International Conference on Financial Cryptography and Data Security*, pages 477–497. Springer, 2017.

[321] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 279–296. IEEE, 2018.

[322] Adil Ahmad, Kyungtae Kim, Muhammad Ihsanulhaq Sarfaraz, and Byoungyoung Lee. Obliviate: A data oblivious filesystem for intel sgx. In *NDSS*, 2018.

[323] Shun Takagi, Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Asymmetric differential privacy. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 1576–1581, 2022.

[324] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. Desire: A third

way for a european exposure notification system leveraging the best of centralized and decentralized systems. *arXiv preprint arXiv:2008.01621*, 2020.

[325] Benny Pinkas and Eyal Ronen. Hashomer-a proposal for a privacy-preserving bluetooth based contact tracing scheme for hamagen, 2020.

[326] Diganth Raj Sehgal. Unpacking the privacy concerns of aarogya setu app, 2020. `https://blog.ipleaders.in/unpacking-privacy-concerns-aarogya-setu-app/`.

[327] David Culler, Prabal Dutta, Gabe Fierro, Joseph E Gonzalez, Nathan Pemberton, Johann Schleier-Smith, Kalyanaraman Shankari, Alvin Wan, and Thomas Zachariah. Covista: A unified view on privacy sensitive mobile contact tracing effort. *arXiv preprint arXiv:2005.13164*, 2020.

[328] Y Luo, N Tang, G Li, W Li, T Zhao, and X Yu. Deepeye: a data science system for monitoring and exploring covid-19 data. *IEEE Data Eng. Bull*, 12, 2020.

[329] Nadeem Ahmed, Regio A Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, and Sanjay K Jha. A survey of covid-19 contact tracing apps. *IEEE Access*, 8:134577–134601, 2020.

[330] enigma, 2020. `https://www.enigma.co/products/`.

[331] Houtan Shirani-Mehr, Farnoush Banaei Kashani, and Cyrus Shahabi. Efficient reachability query evaluation in large spatiotemporal contact datasets. *arXiv preprint arXiv:1205.6696*, 2012.

[332] Elena V. Strzheletska and Vassilis J. Tsotras. Efficient processing of reachability queries with meetings. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '17, New York, NY, USA, 2017. Association for Computing Machinery.

[333] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. Press: A novel framework of trajectory compression in road networks. *arXiv preprint arXiv:1402.1546*, 2014.

[334] Chao Chen, Yan Ding, Xuefeng Xie, Shu Zhang, Zhu Wang, and Liang Feng. Trajcompressor: An online map-matching-based trajectory compression framework leveraging vehicle heading direction and change. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):2012–2028, 2019.

[335] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. Rest: A reference-based framework for spatio-temporal trajectory compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2797–2806, 2018.

List of Publications

- **International Journal Papers**

  [a] **(Chapter 3) Fumiyuki Kato**, Li Xiong, Shun Takagi, Yang Cao, Masatoshi Yoshikawa. ULDP-FL: Federated Learning with Across Silo User-Level Differential Privacy. Submitted to VLDB 2024, **Under Submission**. arXiv preprint arXiv:2308.12210 (2023).

  [b] **(Chapter 4) Fumiyuki Kato**, Yang Cao and Masatoshi Yoshikawa. OLIVE: Oblivious Federated Learning on Trusted Execution Environment against the risk of sparsification. Proc. VLDB Endow. 16, 10 (June 2023), 2404–2417.

  [c] **(Appendix A) Fumiyuki Kato**, Tsubasa Takahashi, Shun Takagi, Yang Cao, Seng Pei Liew, and Masatoshi Yoshikawa. HDPView: differentially private materialized view for exploring high dimensional relational data. Proc. VLDB Endow. 15, 9 (May 2022), 1766–1778.

  [d] **(Appendix B) Fumiyuki Kato**, Yang Cao and Masatoshi Yoshikawa. PCT-TEE: Trajectory-based Private Contact Tracing System with Trusted Execution Environment. ACM Trans. Spatial Algorithms Syst. 8, 2, Article 13 (June 2022), 35 pages.

- **International Conference Papers**

  [e] **(Chapter 5) Fumiyuki Kato**, Yang Cao and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. Data and Applications Security and Privacy XXXV: 35th Annual IFIP WG 11.3 Conference (DBSec), DBSec 2021, Calgary, Canada, July 19–20, 2021, Proceedings 35. Springer

International Publishing, 2021.

[f] **(Appendix B) Fumiyuki Kato**, Yang Cao and Masatoshi Yoshikawa. Secure and efficient trajectory-based contact tracing using trusted hardware. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 4016-4025). IEEE. 7th International Workshop on Privacy and Security of Big Data.

- **Other co-authored Papers**

[g] Shun Takagi, Li Xiong, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa. HRNet: Differentially Private Hierarchical and Multi-Resolution Network for Human Mobility Data Synthesization. Submitted to VLDB 2024, **Under Submission**.

[h] Shun Takagi, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa. From Bounded to Unbounded: Privacy Amplification via Shuffling with Dummies. In 2023 IEEE 36th Computer Security Foundations Symposium (CSF) (pp. 457-472). IEEE.

[i] Shun Takagi, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa. Asymmetric Differential Privacy. In 2022 IEEE International Conference on Big Data (Big Data) (pp. 1576-1581). IEEE.

[j] Ruixuan Cao, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa. An Accurate, Flexible and Private Trajectory-based Contact Tracing System. Information Integration and Web Intelligence (iiWAS). iiWAS 2022. Lecture Notes in Computer Science, vol 13635. Springer, Cham.

[k] Seng Pei Liew, Tsubasa Takahashi, Shun Takagi, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa. Network Shuffling: Privacy Amplification via Random Walks. Proceedings of the 2022 International Conference on Management of Data (SIGMOD). 2022.

[l] Zheng Wen, Keping Yu, Xin Qi, Toshio Sato, Yutaka Katsuyama, Takuro Sato, Wataru Kameyama, **Fumiyuki Kato**, Yang Cao, Masatoshi Yoshikawa, Min Luo and Jun Hashimoto. Blockchain-empowered Contact Tracing for COVID-19 Using Crypto-spatiotemporal Information. 2020 IEEE International Conference on E-health Networking, Application & Services (HEALTHCOM), Shenzhen, China, 2021, pp. 1-6.

Appendix

We have completed individual in-depth studies on DP (Appendix A) and TEE (Appendix B), respectively. Both of them are the key elemental technologies of private and secure federated learning as we consistently argue in this thesis. Here, we present the researches in detail and encourage readers to acquire a better comprehension of actual application and practice of DP and TEE, which clarifies the original motivation and significance of them.

A HDPView: differentially private materialized view for exploring high dimensional relational data

A.1 Introduction

In the early stage of data science workflows, exploring a database to understand its properties in terms of multiple attributes is essential to designing the subsequent tasks. To understand the properties, data analysts need to issue a wide variety of range counting queries. If the database is highly sensitive (e.g., personal healthcare records), data analysts may have little freedom to explore the data due to privacy issues [236, 237].

How can we explore the properties of high-dimensional sensitive data while preserving privacy? This work focuses on guaranteeing DP [42, 141] via random noise injections. As Figure A.1 shows, we especially study how to construct a *privacy-preserving materialized view* (*p-view* for short) of relational data, which enables data analysts to explore arbitrary range counting queries in a differen-

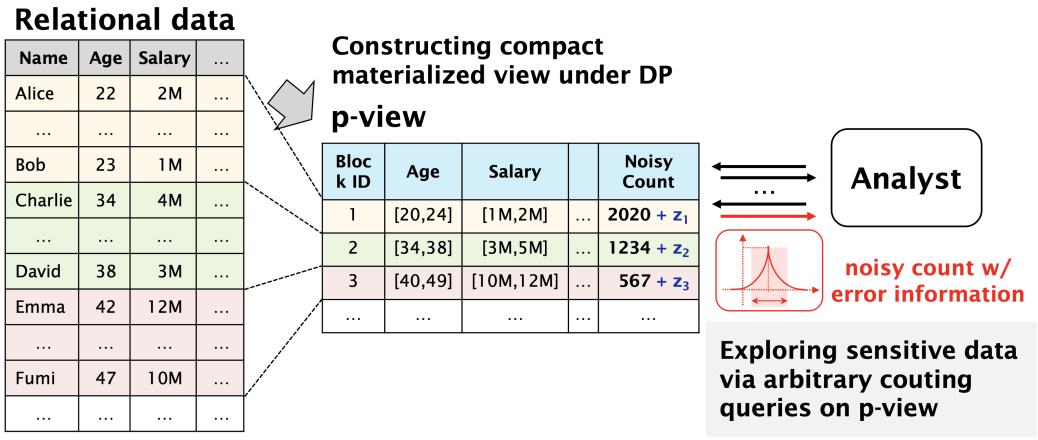


Figure A.1: Data exploration through a *privacy-preserving materialized view* (*p-view* for short) of a multidimensional relational data. The p-view works as an independent query system. Analysts can explore sensitive and multidimensional data by issuing any range counting queries over the p-view before downstream data science workflows.

Table A.1: Only the proposed method achieves all requirements in private data exploration for high-dimensional data. Each competitor represents a baseline, data partitioning, workload optimization, and generative model, respectively.

	Identity [42]	Privtree [114]	HDMM [113]	PrivBayes [242]	HDPView (mine)
Workload independence	✓	✓		✓	✓
Analytical reliability	✓	✓	✓		✓
Noise resistance on high-dimensional		only low-dimensional	✓	✓	✓
Space efficiency			✓	✓	✓

tial private way. Note that once a p-view is created, the privacy budget is not consumed any more for publishing counting queries, different from interactive differentially private query systems [46, 238, 239, 240, 241], which consume the budget every time queries are issued. In this work, we describe the desirable properties of the p-view, especially in data exploration for high-dimensional data, and fill the gaps of the existing methods.

Several methods for constructing a p-view have been studied in the existing literature. The most primitive method is to add Laplace noise [42] to each cell of the count tensor (or vector) representing the original histogram and publish the perturbed data as a p-view. While this noisy view can answer arbitrary range counting queries with a DP guarantee, it accumulates a large amount of

noise. Data-aware partitioning methods [243, 244, 245, 246, 114, 247] are potential solutions, but they focus only on low-dimensional data due to the high complexity of discovering the optimal partitioning when the data have multiple attributes. Additionally, these methods require exponentially large spaces as the dimensionality of the data increases due to the count tensor representation, which can easily make them impractical. Workload-aware optimization methods [248, 249, 113, 247] are promising techniques for releasing query answers for high-dimensional data; however, they cannot provide query-independent p-views needed in data exploration.

In addition, one of the most popular approaches these days is differentially private learning of generative models [250, 251, 242, 117, 115, 252, 253, 116, 254]. Through the training of deep generative models [117, 115, 252, 254, 253] or graphical models [242, 116], counting queries and/or marginal queries can be answered directly from the model or indirectly with synthesized data via sampling. These methods are very space efficient because the synthetic dataset or graphical model can be used to answer arbitrary counting queries. However, these families rely on complex optimization methods such as DP-SGD [48], and it is very difficult to quantitatively estimate the error of counting queries using synthetic data, which eventually leads to a lack of reliability in practical use. Unlike datasets often used in the literature, the data collected in the practical field may be completely unmodelable. Table A.1 summarizes a comparison between the most related works and our method, HDPView, i.e., **H**igh-**D**imensional **P**rivate materialized **View**. Each method is described in more detail in Section A.2.

Our target use case is privacy-preserving data exploration on high-dimensional data, for which the p-view should have the following four properties:

- **Workload independence**: Data analysts desire to issue arbitrary queries for exploring data. These queries should not be predefined.
- **Analytical reliability**: For reliability in practical data exploration, it is necessary to be able to estimate the scale of the error for arbitrary counting queries.
- **Noise resistance on high-dimensional data**: Range counting queries

[a]Identity adds noise to the entries of the count vector by the Laplace mechanism [42] and cannot directly perturb high-dimensional datasets due to the domains being too large; we measure estimated error using the method described in [113].

Table A.2: HDPVIEW provides low-error counting queries in average on various workloads and datasets, and high space-efficiency of privacy-preserving materialized view (p-view) when $\epsilon = 1.0$. (N/A is due to HDMM and PrivBayes do not create p-view.)

	Identity [42]	Privtree [114]	HDMM [113]	PrivBayes [242]	HDPVIEW (ours)
Average relative RMSE	1.94×10^7	7.05	35.34	3.79	**1.00**
Average relative size of p-view	4.59×10^{17}	5578.27	N/A	N/A	**1.00**

compute the sum over the range and accumulate the noise injected for DP. This noise accumulation makes the query answers useless. To avoid this issue, we need a robust mechanism even for high-dimensional data.

- **Space efficiency**: It is necessary to generate spatially efficient views even for count tensors with a large number of total domains on various datasets.

Our proposal. To satisfy all the above requirements, we propose a simple yet effective recursive bisection method on a high-dimensional count tensor, HDPVIEW. The proposed method has the same principle as [245, 248, 114] of first partitioning a database into small blocks and then averaging over each block with noise. Unlike the existing methods, HDPVIEW can efficiently perform error-minimizing partitioning even on multidimensional count tensors instead of conventional 1D count vectors. HDPVIEW recursively partitions multidimensional blocks at a cutting point chosen in a differentially private manner while aiming to minimize the sum of aggregation error (AE) and perturbation error (PE). Compared to Privtree [114], our proposed method provides a more data-aware flexible cutting strategy and proper convergence of block partitioning, which results in smaller errors in private counting queries and much better spatial efficiency of the generated p-views. Our method provides a powerful and practical solution for constructing a p-view under the DP constraint. More importantly, the p-view generated by HDPVIEW can work as a query processing system and expose the estimated error bound at runtime for any counting query without further privacy consumption. This error information ensures reliable analysis for data explorers.
Contributions. Our contributions are threefold. First, we design a *p-view* and formalize the segmentation for a multidimensional count tensor to find an effective p-view as error minimizing optimization problem. P-view can be widely used for data exploration process on multidimensional data and is a differentially private

approximation of a multidimensional histogram that can release counting queries with analytical reliability. Second, we propose HDPView described above to find a desirable solution to the optimization problem. Our algorithm is more effective than conventional algorithms due to finding flexible partitions and more efficient due to making appropriate convergence decisions. Third, we conduct extensive experiments, whose source code and dataset are open, and show that HDPView has the following merits. (1) Effectiveness: HDPView demonstrates smaller errors for various range counting queries and outperforms the existing methods [42, 114, 248, 113, 242] on multi-dimensional real-world datasets. (2) Space efficiency: HDPView generates a much more compact representation of the p-view than the state-of-the-art (i.e., Privtree [114]) in the experiment.

Preview of result. We present a summary previewing of the experimental results. Table A.2 shows the average relative root mean squared error against (RMSE) of HDPView in eight types of range counting queries on eight real-world datasets and the average relative size of the p-view generated by the algorithms. With Identity, we obtain a p-view by making each cell of the original count tensor a converged block. [a] HDPView yields the smallest error score on average. Several algorithms show better results than HDPView in some queries, but HDPView is the best on average. This is a desirable property for data explorations. Furthermore, compared to that of Privtree [114], the p-view generated by HDPView is more space efficient.

We believe that our proposed method could help data analysts explore sensitive data in the early stages of data mining pipelines while preserving data utility and privacy. Therefore, in a practical data science workflow, our proposed method is useful to design the workload in-house, before applying the workload-aware query processing to release the private results. Furthermore, the advantage that our method can release an unlimited number of arbitrary queries without additional privacy budget is useful for providing an interface for interactive queries to third parties. Most of existing DP-query engines [46, 241, 238] consider privacy consumption by sequential composition or advanced sub-linear composition [240], which results in restricting data explorations.

[a] Note, the scores of DAWA and Identity are only on low-dimensional datasets.

A.2 Related Works

In the last decade, several works have proposed differentially private methods for exploring sensitive data. Here, we describe the state-of-the-arts related to our work.

Data-aware partitioning. Data-aware partitioning is a conventional method that aims to directly randomize and expose the entire histogram for all domains (e.g., count vector, count tensor); thereby, it can immediately compose a p-view that answers all counting queries. A naïve approach to constructing a differentially private view is adding Laplace noise [42] to all values of a count vector; this is called the Identity mechanism. This naïve approach results in prohibitive noise on query answers through the accumulation of noise over the grouped bins used by queries. DAWA [248] and AHP [243] take data-aware partitioning approaches to reduce the amount of noise. The partitioning-based approaches first split a raw count vector into bins and then craft differentially private aggregates by averaging each bin and injecting a single unit of noise in each bin. However, these approaches work only for very low (e.g., one or two) -dimensional data due to the high complexity of discovering the optimal solution when the data have multiple attributes. DPCube [245] is a two-step multidimensional data-aware partitioning method, but the first step, obtaining an accurate approximate histogram, is difficult on high-dimensional data with small counts in each cell.

Privtree [114] and [255] perform multidimensional data-aware partitioning on count tensors, mainly targeting the spatial decomposition task for spatial data. Unlike our method, this method uses a fixed quadtree as the block partitioning strategy, which leads to an increase in unnecessary block partitioning as the dimensionality increases. As a result, it downgrades the spatial efficiency and incurs larger perturbation noise. In addition, this method aims to partition the blocks such that the count value is below a certain threshold, while our proposed method aims to minimize the AE of the blocks and reduce count query noise.

Optimization of given workloads. Another well-established approach is the optimization for a given workload. Li et al. [249] introduced a matrix mechanism (MM) that crafts queries and outputs optimized for a given workload. The high-dimensional MM (HDMM) [113] is a workload-aware data processing method extending the MM to be robust against noise for high-dimensional data. PrivateSQL [247] selects the view to optimize from pregiven workloads. In the

data exploration process, it is not practical to assume a predefined workload, and these methods are characterized by a loss of accuracy when optimized for a workload of wide variety of queries.

Private data synthesis. Private data synthesis, which builds a privacy-preserving generative model of sensitive data and generates synthetic records from the model, is also useful for data exploration. Note that synthesized dataset can work as a p-view by itself. PrivBayes [242] can heuristically learn a Bayesian network of data distribution in a differentially private manner. DPPro [256], Priview [250] and PrivSyn [116] represent distribution by approximation with several smaller marginal tables. While these methods provide a partial utility guarantee based on randomized mechanisms such as Laplace mechanisms or random projections, they face difficulties in providing an error bound for arbitrary counting queries on the synthesized data. Differentially private deep generative models have also attracted attention [257, 117, 115, 252], but most of the works focus on the reconstruction of image datasets. Fan et al. [253] studied how to build a good generative model based on generative adversarial nets (GAN) for tabular datasets. Their experimental results showed that the utility of differentially private GAN was lower than that of PrivBayes for tabular data. [258] provides a solution for high-dimensional data in a local DP setting.

As mentioned in Section A.1, the accuracy of these methods has improved greatly in recent years, but it is difficult to guarantee their utility for analysis using counting queries, and there are large gaps in practice. DPPro [256] utilizes a random projection [259] that preserve L2-distance to the original data in an analyzable form to give a utility guarantee, but this is different from the guarantee for each counting query. CSM [260] gives a utility analysis for queries, however, their analysis ignores the effect of information loss due to compression, which may not be accurate. Also, as shown in their experiments, they apply intense preprocessing to the domain size and do not show the effectiveness for high-dimensional data. Our proposed method provides an end-to-end error analysis for arbitrary counting queries by directly constructing p-views from histograms without any intermediate generative model.

Querying and programming framework. PrivateSQL [247] is a differentially private relational database system for multirelational tables, where for each table, it applies an existing noise-reducing method such as DAWA. Unlike our method, PrivateSQL needs a given workload to design private views to release. Flex [46],

Google's work [239] and APEx [238] are SQL processing frameworks under DP. They issue queries to the raw data, which can consume an infinite amount of the privacy budgets. Hence, we believe that these DP-query processing engines are not suitable for data exploration tasks where many instances of trial and error may be possible. Our method generates p-view, which can be used as a differentially private query system that allows any number of range counting queries to be issued.

A.3 Preliminaries

This section introduces essential knowledge for understanding our proposal. We first describe notations this work uses. Then, we briefly explain DP.

Notation

Let X be the input database with n records consisting of an attribute set A that has d attributes $A = \{a_1, \ldots, a_d\}$. The domain $dom(a)$ of an attribute a has a finite ordered set of discrete values, and the size of the domain is denoted as $|dom(a)|$. The overall domain size of A is $|dom(A)| = \prod_{i \in [d]} |dom(a_i)|$, where $[d] = \{1, \ldots, d\}$. In the case where attribute a is continuous, we transform the domain into a discrete domain by binning, and in the case where attribute a is categorical, we transform it into an ordered domain. Then, $dom(a)$ can be represented as a *range* $r[s_a, e_a]$ where for all $p_a \in dom(a)$, $s_a \leq p_a \leq e_a$. For ranges r_1, r_2, $|r_1 \cap r_2|$ means the number of value p_a satisfies $s_a \leq p_a \leq e_a$.

We consider transforming the database X into the d-mode count tensor $\mathcal{X}$, where given d ranges $r_1, \ldots, r_d$, $\mathcal{X}[r_1, \ldots, r_d]$ represents the number of records where $(a_1(\in r_1), \ldots, a_d(\in r_d)) \in X$. We utilize x $(\in \mathcal{X})$ as a count value in $\mathcal{X}$; this corresponds to a cell of the count tensor. We denote a subtensor of $\mathcal{X}$ as *block* $\mathcal{B} \subseteq \mathcal{X}$. $\mathcal{B}$ is also a d-mode count tensor, but its domain in each dimension is smaller than or equal to that of the original count tensor $\mathcal{X}$; i.e., each attribute a_i $(i \in [d])$, $r[s_{a_i}, e_{a_i}]$ of $\mathcal{B}$ and $r[s'_{a_i}, e'_{a_i}]$ of $\mathcal{X}$ satisfy $s'_{a_i} \leq s_{a_i}$ and $e_{a_i} \leq e'_{a_i}$. We denote the domain size of $\mathcal{B}$ as $|\mathcal{B}|$.

Last, we denote q as a counting query and $\mathbf{W}$ as a workload. $\mathbf{W}$ is a set of $|\mathbf{W}|$ counting queries, where $\mathbf{W} = \{q_1, ..., q_{|\mathbf{W}|}\}$, and $q(\mathcal{X})$ returns the counting query results for count tensor $\mathcal{X}$.

Basic DP Mechanisms

The Laplace mechanism and exponential mechanism are well-known as standard approaches to satisfy DP. The Laplace mechanism can be used for randomizing numerical data as shown in Definition 3. Releasing the histogram is a typical use case of this mechanism. The exponential mechanism is the random selection algorithm. The selection probability is weighted based on a score in a quality metric for each item.

Definition 12 (Exponential Mechanism)**.** *Let q be the quality metric for choosing an item $y \in Y$ in the database D. The exponential mechanism randomly samples y from Y with weighted sampling probability defined as follows:*

$$\Pr[y] \sim \exp(\frac{\epsilon q(D, y)}{2\Delta_q}). \tag{7.1}$$

A.4 Problem Formulation

Segmentation as Optimization

This section describes the foundation of multidimensional data-aware segmentation that seeks a solution for the differentially private view $\tilde{\mathcal{X}}$ from the input count tensor $\mathcal{X}$. Every count $\tilde{x} \in \tilde{\mathcal{X}}$ is sanitized to satisfy DP. We formulate multidimensional block segmentation as an optimization problem.

Foundation. Given a count tensor $\mathcal{X}$, we consider partitioning $\mathcal{X}$ into m blocks $\pi = \{\mathcal{B}_1, ..., \mathcal{B}_m\}$. The blocks satisfy $\mathcal{B}_i \cap \mathcal{B}_j = \emptyset$ where $i, j \in [m]$, $j \neq i$ and $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_m = \mathcal{X}$. We denote the sum over $\mathcal{B}_i$ as $S_i = \sum_{x' \in \mathcal{B}_i} x'$ and its perturbed output as $\tilde{S}_i = S_i + z_i$. We can sample z_i with the Laplace mechanism $Lap(1/\epsilon)$ and craft the ϵ-differentially private sum in $\mathcal{B}_i$.

For any count x in the block $\mathcal{B}_i$, we have two types of errors: *Perturbation Error* (PE) and *Aggregation Error* (AE). Assuming that we replace any count $x \in \mathcal{B}_i$ with $\bar{x}_i = (S_i + z_i)/|\mathcal{B}_i|$, the absolute error between x and $\bar{x}_i$ can be computed as

$$|x - \bar{x}_i| = \left| \left(x - \frac{S_i}{|\mathcal{B}_i|} \right) - \frac{z_i}{|\mathcal{B}_i|} \right| \leq \left| x - \frac{S_i}{|\mathcal{B}_i|} \right| + \left| \frac{z_i}{|\mathcal{B}_i|} \right|. \tag{7.2}$$

Therefore, the total error over block $\mathcal{B}_i$, namely, the segmentation error (SE), can be given by:

$$\mathrm{SE}(\mathcal{B}_i) = \sum_{x \in \mathcal{B}_i} |x - \bar{x}_i| \leq \mathrm{AE}(\mathcal{B}_i) + \mathrm{PE}(\mathcal{B}_i) \tag{7.3}$$

where

$$\mathrm{AE}(\mathcal{B}_i) \coloneqq \sum_{x \in \mathcal{B}_i} \left| x - \frac{S_i}{|\mathcal{B}_i|} \right|, \tag{7.4}$$

$$\mathrm{PE}(\mathcal{B}_i) \coloneqq |z_i| \,. \tag{7.5}$$

(7.4) and (7.5) represent the AE and the PE, respectively.

Problem. The partitioning makes the PE of each block $\frac{1}{|\mathcal{B}_i|}$ times smaller than those of the original counts with Laplace noise. Furthermore, we consider the expectation of the SE

$$\begin{aligned} \mathbb{E}\left[\sum_{i\in[m]} \mathrm{SE}(\mathcal{B}_i)\right] &\leq \mathbb{E}\left[\sum_{i\in[m]} \mathrm{AE}(\mathcal{B}_i)\right] + \mathbb{E}\left[\sum_{i\in[m]} \mathrm{PE}(\mathcal{B}_i)\right] \\ &= \sum_{i\in[m]} \mathrm{AE}(\mathcal{B}_i) + \sum_{i\in[m]} \mathbb{E}\left[\mathrm{PE}(\mathcal{B}_i)\right] \\ &= \sum_{i\in[m]} \mathrm{AE}(\mathcal{B}_i) + m \cdot \frac{1}{\epsilon}. \end{aligned} \tag{7.6}$$

Thus, to discover the optimal partition π, we need to minimize Eq. (7.6). The optimization problem is denoted as follows:

$$\begin{aligned} &\underset{\pi}{\text{minimize}} \sum_{\mathcal{B}_i \in \pi} \left(\mathrm{AE}(\mathcal{B}_i) + \frac{1}{\epsilon} \right) \\ &\text{subject to } \mathcal{B}_i \cap \mathcal{B}_{j\neq i} = \emptyset, \quad \mathcal{B}_i, \mathcal{B}_j \in \pi \\ &\qquad \bigcup_{\mathcal{B}_i \in \pi} \mathcal{B}_i = \mathcal{X} \end{aligned} \tag{7.7}$$

Challenges. It is not easy to discover the optimal partition π. This problem is an instance of the set partitioning problem [261], which is known to be NP-complete, where the objective function is computed by brute-force searching for every combination of candidate blocks. It is hard to solve since the search space is basically a very large scale due to large $|dom(A)|$. Therefore, this work seeks an efficient heuristic solution with a good balance between utility (i.e., smaller errors) and privacy.

P-view Definition

Our proposed p-view has a simple structure. The p-view consists of a set of blocks, each of which has a range for each attribute and an appropriately randomized

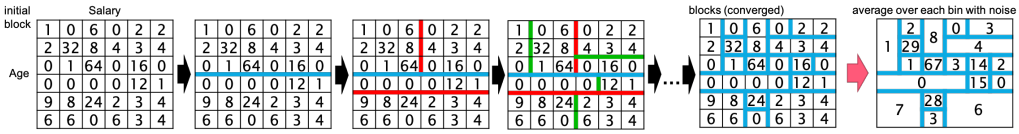


Figure A.2: HDPView efficiently discovers blocks (i.e., groups of count cells) with smaller AEs (black arrow) and averages over each block with injected noise (red arrow). The p-view stores the randomized counts in a blockwise way.

count value, as shown in Figure A.1. Formally, we define the p-view as follows.

$$\begin{aligned} \text{p-view } \tilde{\mathcal{X}} &= \{\mathcal{B}_1, ..., \mathcal{B}_m\}, \\ \text{for } i \in [m], \mathcal{B}_i &= (\{r[s^{(i)}_{a_1}, e^{(i)}_{a_1}], ..., r[s^{(i)}_{a_d}, e^{(i)}_{a_d}]\}, \tilde{S}_i) \end{aligned} \tag{7.8}$$

Thus, each block $\mathcal{B}_i$ has this d-dimensional domain and the sanitized sum of count values $\tilde{S}_i$.

In the range counting query processing, a counting query q needs to have the range condition $c_q = \{r[s^{(q)}_{a_1}, e^{(q)}_{a_1}], ..., r[s^{(q)}_{a_d}, e^{(q)}_{a_d}]\}$. Let the ranges of $\mathcal{B}_i$ be $\{r[s^{(i)}_{a_1}, e^{(i)}_{a_1}], ..., r[s^{(i)}_{a_d}, e^{(i)}_{a_d}]\}$, and we calculate the intersection of c_q and the block and add the count value according to the size of the intersection. Hence, the result can be calculated as follows.

$$q(\tilde{\mathcal{X}}) = \sum_{i=1,...,m} \left(\prod_{l=1,...,d} \left(\left| r[s^{(q)}_{a_l}, e^{(q)}_{a_l}] \cap r[s^{(i)}_{a_l}, e^{(i)}_{a_l}] \right| \right) * \frac{\tilde{S}_i}{|\mathcal{B}_i|} \right) \tag{7.9}$$

The number of intersection calculations is proportional to the number of blocks, and the complexity of the query processing is $\mathcal{O}(md)$.

A.5 Proposed Algorithm

This section introduces our proposed solution. Our solution constructs a p-view of the input relational data while preserving utility and privacy with analytical reliability to estimate errors in the arbitrary counting queries against the p-view (Eq.(7.9)).

Overview

The challenge is to devise a simple yet effective algorithm that enables us to efficiently search a block partitioning with small total errors and DP guarantees. As a realization of the algorithm, we propose HDPView.

Figure A.2 illustrates an overview of our proposed algorithm. First, HDPView creates the initial block $\mathcal{B}^{(0)}$ that covers the whole count tensor $\mathcal{X}$. Second, we recursively bisect a block $\mathcal{B}$ (initially $\mathcal{B} = \mathcal{B}^{(0)}$) into two disjoint blocks $\mathcal{B}_L$ and $\mathcal{B}_R$. Before bisecting $\mathcal{B}$, we check whether the AE over $\mathcal{B}$ is sufficiently small. If the result of the check is positive, we stop the recursive bisection for $\mathcal{B}$. Otherwise, we continue to split $\mathcal{B}$. We pick a splitting point $p \in dom(a)$ $(a \in A)$ for splitting $\mathcal{B}$ into $\mathcal{B}_L$ and $\mathcal{B}_R$ which have smaller AEs. Although splitting does not always result in smaller total AEs, proper cut point obviously makes AEs much smaller. Third, HDPView recursively executes these steps separately for $\mathcal{B}_L$ and $\mathcal{B}_R$. After convergence is met for all blocks, HDPView generates a randomized aggregate by $S_i + z_i$ where $z_i \sim Lap(1/\epsilon)$ for each block $\mathcal{B}_i$. Finally, for all $x \in \mathcal{B}_i$, we obtain the randomized count $\tilde{x} = (S_i + z_i)/|\mathcal{B}_i|$.

The above-mentioned algorithm can discover blocks that heuristically reduce the AEs, and is efficient due to its simplicity. However, the question is *how can we make the above algorithm differentially private?*. To solve this question, we introduce two mechanisms, *random converge* (Section A.5) and *random cut* (Section A.5). Random converge determines the convergence of the recursive bisection, and random cut determines the effective single cutting point. These provide reasonable partitioning strategy to reduce the total errors with small privacy budget consumption.

The overall algorithm of HDPView are described in Algorithm 16. Let $\epsilon_b = \epsilon_r + \epsilon_p$ be the total privacy budget for HDPView, where ϵ_r is the budget for the recursive bisection and ϵ_p is the budget for the perturbation. HDPView utilizes $\gamma\epsilon_r$ for random converge and $(1-\gamma)\epsilon_r$ for random cut $(0 \leq \gamma \leq 1)$. α is a hyperparameter that determines the size of λ and δ, where λ corresponds to the Laplace noise scale of random converge (Lines 8, 17) and δ is a bias term for AE (Lines 9, 16). These are, sketchily, tricks for performing random converge with depth-independent scales, which are explained in Section A.5 and a detailed proof of DP is given in Section A.5. The algorithm runs recursively (Lines 10, 32, 33), alternating between random converge (Lines 15-19) and random cut (Lines 21-30). The random converge stops when the AE becomes small enough, consuming a total budget of $\gamma\epsilon_r$ independent of the number of depth. The random cut consumes a budget of $(1-\gamma)\epsilon_r/\kappa$ for each cutting point selection until the depth exceeds κ. κ is set as $\kappa = \beta \log_2 \bar{n}$, where $\beta > 0$ is hyperparameter and $\bar{n}$ is the total domain size of the data. As we see later in Theorem 9, AE is not increased

by splitting, so if the depth is greater than κ, we split randomly without any privacy consumption until convergence. After the recursive bisection converges, HDPView perturbs the count by adding the Laplace noise while consuming ϵ_p.

Random Converge

AE decreases by properly splitting the blocks, however unnecessary block splitting leads to an increase in PE as mentioned above. To stop the recursive bisection at the appropriate depth, we need to obtain the exact AE of the block, which is a data-dependent output, therefore we need to provide a DP guarantee. One approach is to publish differential private AE so that making the decision for the stop is also DP by the post-processing property. In other words, the stop is determined by $\mathrm{AE}(\mathcal{B}) + Lap(\lambda) \leq \theta$ where θ is a threshold indicating AE is small enough. However, this method consumes privacy budget every time the AE is published, and the budget cannot be allocated unless the depth of the partition is decided in advance. Therefore, we utilize the observation for the *privacy loss* of Laplace mechanism-based threshold query [114] and design the biased AE (BAE) of the block $\mathcal{B}$ instead of $\mathrm{AE}(\mathcal{B})$: $\mathrm{BAE}(\mathcal{B}) = \max(\theta+2-\delta,\ \mathrm{AE}(\mathcal{B})-k\delta)$, where k is the current depth of bisection, δ (> 0) is a bias parameter, i.e., we determine the convergence by $\mathrm{BAE}(\mathcal{B}) + Lap(\lambda) \leq \theta$. Intuitively, the BAE is designed to tightly bound the privacy loss of the any number of Laplace mechanism-based threshold queries with constant noise scale λ. When the value is sufficiently larger than the threshold, this privacy loss decreases exponentially [114]. Then, it can be easily bounded by an infinite series regardless of the number of queries. Conversely, when the value is small compared to the threshold, each threshold query consumes a constant budget. To limit the number of such budget consumptions, a bias δ is used to force a decrease in the value for each threshold query (i.e., each depth) because BAE has a minimum and if the value is guaranteed to be less than the minimum for adjacent databases, the privacy loss is zero. The design of our BAE allows for two constant budget consumptions at most, with the remainder being bounded by an infinite series. We give a detailed proof in Section A.5. As a whole, since BAE is basically close to AE, AEs are expected to become sufficiently small overall.

Then, we consider about θ where if θ is too large, block partitioning will not sufficiently proceed, causing large AEs, and if it is too small, more blocks will be generated, leading to increase in total PEs. To prevent unwanted splitting, it is

Algorithm 16 HDPView

Input: initial block $\mathcal{B}^{(0)}$, privacy budget ϵ_b, recursive bisection budget ratio ϵ_r/ϵ_b, hyperparameters α, β, γ
Output: p-view $\tilde{\mathcal{X}}$

procedure HDPView($\mathcal{B}^{(0)}, \epsilon_b, \epsilon_r/\epsilon_b, \alpha, \beta, \gamma$)
  $\epsilon_r \leftarrow \epsilon_b \cdot (\epsilon_r/\epsilon_b)$; $\epsilon_p \leftarrow \epsilon_b \cdot (1 - \epsilon_r/\epsilon_b)$
  $\tilde{n} \leftarrow$ TotalDomainSizeOf($\mathcal{X}$)
  $\kappa \leftarrow \beta \log_2 \tilde{n}$ // maximum depth of random cut
  $\pi \leftarrow \{\}$; $k \leftarrow 1$ // converged blocks; current depth
  $\theta \leftarrow 1/\epsilon_p$ // threshold
  $\epsilon_{cut} \leftarrow (1-\gamma)\epsilon_r/\kappa$ // privacy budget for random cut
  $\lambda \leftarrow \left(\frac{2\alpha-1}{\alpha-1} + 1\right) \cdot \left(\frac{2}{\gamma\epsilon_r}\right)$ // noise scale for random converge
  $\delta \leftarrow \lambda \log \alpha$ // bias parameter
  RecursiveBisection($\mathcal{B}^{(0)}, \pi, \epsilon_{cut}, k, \kappa, \theta, \lambda, \delta$)
  $\tilde{\mathcal{X}} \leftarrow$ Perturbation(π, ϵ_p)
  return $\tilde{\mathcal{X}}$
end procedure
procedure RecursiveBisection($\mathcal{B}$, π, ϵ_{cut}, k, κ, θ, λ, δ)
  /* Random Converge */
  BAE($\mathcal{B}$) $\leftarrow \max(\theta + 2 - \delta,\ \text{AE}(\mathcal{B}) - k\delta)$
  if BAE($\mathcal{B}$) + Lap(λ) $\leq \theta$ **then**
    $\pi \leftarrow \pi \bigcup \mathcal{B}$
    return
  end if
  /* Random Cut */
  if $k \leq \kappa$ **then**
    for all $i \in [d], j \in [|dom(a_i)|]$ **do**
      $quality[i,j] \leftarrow Q(B, a_{ij})$
    end for
    $(i^*, j^*) \leftarrow$ WeightedSampling($\epsilon_{cut}, quality$)
  else
    $(i^*, j^*) \leftarrow$ RandomSampling($[d], [|dom(a_i)|]$)
  end if
  $(\mathcal{B}_L, \mathcal{B}_R) \leftarrow$ Split(i^*, j^*)
  /* Repeat Recursively */
  RecursiveBisection($\mathcal{B}_L, \pi, \epsilon_{cut}, k+1, \kappa, \theta, \lambda, \delta$)
  RecursiveBisection($\mathcal{B}_R, \pi, \epsilon_{cut}, k+1, \kappa, \theta, \lambda, \delta$)
  return
end procedure

appropriate to stop when the increase in PE is greater than the decrease in AE. We design the threshold θ as $1/\epsilon_p$ which is the standard deviation of the Laplace noise to be perturbed. Considering the each bisection increases the total PE by $1/\epsilon_p$, when the AE becomes less than the PE, the division will increase the error at least. Hence, it is reasonable to stop under this condition.

Random Cut

Here, the primary question is how to pick a reasonable cutting point from all attribute values in a block $\mathcal{B}$ under DP. The intuition is that a good cutting point results in smaller AEs in the two split blocks. We design random cut by combining an exponential mechanism with scoring based on the total AE after splitting.

Let $\mathcal{B}_L^{(p)}$ and $\mathcal{B}_R^{(p)}$ be the blocks split from $\mathcal{B}$ by the cutting point p, and the quality function Q of p in $\mathcal{B}$ is defined as follows:

$$Q(\mathcal{B}, p) = -(\mathrm{AE}(\mathcal{B}_L^{(p)}) + \mathrm{AE}(\mathcal{B}_R^{(p)})). \tag{7.10}$$

Then, we compute the score for all attribute values $p \in dom(a)$, $a \in A$, and satisfies $|\mathcal{B}_L^{(p)}| \geq 1$ and $|\mathcal{B}_R^{(p)}| \geq 1$. Note that the number of candidates for p is proportional to the sum of the domains for each attribute $\sum_{i\in[d]} |dom(a_i)|$, not to the total domains $\prod_{i\in[d]} |dom(a_i)|$. We employ weighted sampling via an exponential mechanism to choose one cutting point $p*$. The sampling probability of p is proportional to

$$\Pr[p* = p] \sim \exp\left(\frac{\epsilon Q(B, p)}{2\Delta_Q}\right) \tag{7.11}$$

where Δ_Q is the L1-sensitivity of the quality metric Q. We denote the L1-sensitivity of AE as Δ_{AE}, and we can easily find $\Delta_Q = 2\Delta_{AE}$ because Q is the sum of two AEs. Thus, each time a cut point p is published according to such weighted sampling, a privacy budget of ϵ is consumed. We set ϵ as the budget allocated to random cut (i.e., $(1-\gamma)\epsilon_r$) divided by κ. If the cutting depth exceeds κ, we switch to random sampling (Line 28 in Algorithm 16). Hence, cutting will not stop regardless of the depth or budget.

Compared to Privtree [114], for a d-dimensional block, at each cut, HDPView generates just 2 blocks with this random cut while Privtree generates 2^d blocks with fixed cutting points. Privtree's heuristics prioritizes finer partitioning, which

sufficiently works in low-dimensional data because AEs become very small and the total PEs is not so large. In high-dimensional data, however, it causes unnecessary block splitting resulting in too much PEs. HDPView carefully splits blocks one by one, thus suppressing unnecessary block partitioning and reducing the number of blocks i.e., smaller PEs. It also enables flexibly shaped multidimensional block partitioning. Moreover, while whole design of HDPView including convergence decision logic and cutting strategy are based on an error optimization problem as described in Section A.4, Privtree has no such background. This allows HDPView to provide effective block partitioning rather than simply fewer blocks, which we empirically confirm in Section A.6.

Privacy Accounting

For privacy consumption accounting, since HDPView recursively splits a block into two disjoint blocks, we only have to trace a path toward convergence. In other words, because HDPView manipulates all the blocks separately, we can track the total privacy consumption by the parallel composition for each converged block. The information published by the recursive bisection is the result of segmentation; however, note that since there is a constraint on the cutting method for the block, it must be divided into two parts; in the worst case, the published blocks may expose all the cutting points. For a given converged block $\mathcal{B}$, we denote the series of cutting points by $S_{\mathcal{B}} = [p_1, ...p_k]$, and $\mathcal{B}_{p_i}$ as the block after being divided into two parts at cutting point p_i. To show the DP guarantee, let D and D' be the neighboring databases, and let $\Pr[S_{\mathcal{B}}|D]$ be the probability that $S_{\mathcal{B}}$ is generated from D. We need to show that for any D, D', and $S_{\mathcal{B}}$ that

$$\left| \frac{\Pr[S_{\mathcal{B}}|D]}{\Pr[S_{\mathcal{B}}|D']} \right| \leq e^{\epsilon} \tag{7.12}$$

to show that the recursive bisection satisfies ϵ-DP.

The block with the largest $\left| \frac{\Pr[S_{\mathcal{B}}|D]}{\Pr[S_{\mathcal{B}}|D']} \right|$ of the converged disjoint blocks is $\mathcal{B}^*$, which has the longest $S_{\mathcal{B}^*}$ and contains different data between D and D'. Random

converge and random cut are represented as follows.

$$\left|\frac{\Pr[S_{\mathcal{B}^*}|D]}{\Pr[S_{\mathcal{B}^*}|D']}\right| = \frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_0}) + Lap(\lambda) > \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}'_{p_0}) + Lap(\lambda) > \theta]} \cdot \frac{\Pr[p* = p_1|D]}{\Pr[p* = p_1|D']} \cdot \frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_1}) + Lap(\lambda) > \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}'_{p_1}) + Lap(\lambda) > \theta]} \cdot \cdots \cdot \frac{\Pr[p* = p_k|D]}{\Pr[p* = p_k|D']} \cdot \frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_k}) + Lap(\lambda) \leq \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}'_{p_k}) + Lap(\lambda) \leq \theta]} \tag{7.13}$$

where $\mathcal{B}_{p_0}$ is the initial count tensor and for all i, $\mathcal{B}'_{p_i}$ indicates a neighboring block for $\mathcal{B}_{p_i}$. Taking the logarithm,

$$\ln\left(\frac{\Pr[S_{\mathcal{B}^*}|D]}{\Pr[S_{\mathcal{B}^*}|D']}\right) = \underbrace{\sum_{i=1}^{k} \ln\left(\frac{\Pr[p* = p_i|D]}{\Pr[p* = p_i|D']}\right)}_{(*1)\ :\ \text{for random cut}} + \underbrace{\sum_{i=0}^{k} \ln\left(\frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_i}) + Lap(\lambda) > \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}'_{p_i}) + Lap(\lambda) > \theta]}\right) + \ln\left(\frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_k}) + Lap(\lambda) \leq \theta|D]}{\Pr[\mathrm{BAE}(\mathcal{B}'_{p_k}) + Lap(\lambda) \leq \theta|D']}\right)}_{(*2)\ :\ \text{for random converge}}. \tag{7.14}$$

and let the first item of the right-hand of Eq.(7.14) be $(*1)$, and the other items be $(*2)$.

$(*1)$ corresponds to the privacy of the random cut, with each probability following Eq.(7.11). Given $\epsilon = \epsilon_{cut}$, for any k, the following holds from sequential composition.

$$\left|\sum_{i=1}^{k} \ln\left(\frac{\Pr[p* = p_i|D]}{\Pr[p* = p_i|D']}\right)\right| \leq \kappa\epsilon_{cut} = (1-\gamma)\epsilon_r. \tag{7.15}$$

The following are privacy guarantees for the other part, $(*2)$, based on the observations presented in [114]. First, we consider the sensitivity of AE Δ_{AE}.

Theorem 8. *The L1-sensitivity of the AE is* $2(1 - 1/|\mathcal{B}|)$.

Proof. Let $\mathcal{B}'$ be the block that differs by only one count from $\mathcal{B}$. The $\mathrm{AE}(\mathcal{B}')$ can be computed as follows:

$$\mathrm{AE}(\mathcal{B}') = \sum_{i \neq j \in [|\mathcal{B}|]} \left|x_i - \frac{S+1}{|\mathcal{B}|}\right| + \left|x_j + 1 - \frac{S+1}{|\mathcal{B}|}\right|.$$

Finally, the L1-sensitivity of AE can be derived as:

$$\Delta_{AE} = (|\mathcal{B}| - 1)\frac{1}{|\mathcal{B}|} + 1 - \frac{1}{|\mathcal{B}|} = 2(1 - 1/n)$$

□

Thus, we also obtain $|\mathrm{BAE}(\mathcal{B}) - \mathrm{BAE}(\mathcal{B}')| \leq 2$, and

$$(*2) \leq \sum_{i=0}^{k-1} \ln\left(\frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_i}) + Lap(\lambda) > \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}_{p_i}) - 2 + Lap(\lambda) > \theta]}\right) + \ln\left(\frac{\Pr[\mathrm{BAE}(\mathcal{B}_{p_k}) + Lap(\lambda) \leq \theta]}{\Pr[\mathrm{BAE}(\mathcal{B}_{p_k}) + 2 + Lap(\lambda) \leq \theta]}\right). \tag{7.16}$$

Furthermore, from the proof in the Appendix in [114], when we have $f(x) = \ln\left(\frac{\Pr[x+Lap(\lambda)>\theta]}{\Pr[x-2+Lap(\lambda)>\theta]}\right)$, then

$$\begin{cases} f(x) \leq \frac{2}{\lambda}, & (\theta - x + 2 > 0) \\ f(x) \leq \frac{2}{\lambda}\exp\left(\frac{\theta - x + 2}{\lambda}\right), & (\theta - x + 2 \leq 0) \end{cases} \tag{7.17}$$

Next, we show the monotonic decreasing property of AE for block partitioning.

Theorem 9. *For any $i = 0, ..., k-1$, $AE(\mathcal{B}_{p_i}) \geq AE(\mathcal{B}_{p_{i+1}})$.*

Proof. We show that when $\mathcal{B}^+$ is an arbitrary block $\mathcal{B}$ with an arbitrary element x (> 0) added to it, the AEs always satisfy $\mathrm{AE}(\mathcal{B}) \leq AE(\mathcal{B}^+)$. Let the elements in $\mathcal{B}$ be $x_1, ..., x_k$, and let $\mathcal{B}^+$ be the block with x_{k+1} added. The mean values in each block are $\bar{x} = \frac{1}{k}(x_1 + \cdots + x_k)$ and $\bar{x}^+ = \frac{1}{k+1}(x_1 + \cdots + x_{k+1})$ and $\mathrm{AE}(\mathcal{B}) = \sum_{i=1}^{k} |x_i - \bar{x}|$ and $\mathrm{AE}(\mathcal{B}^+) = \sum_{i=1}^{k+1} |x_i - \bar{x}^+|$. Considering how much the AE can be reduced with the addition of x_{k+1} to $\mathcal{B}$, $|x_i - \bar{x}| - |x_i - \bar{x}^+| \leq |\bar{x} - \bar{x}^+|$ holds for each i $(= 1, ..., k)$, so $\mathrm{AE}(\mathcal{B}) - \mathrm{AE}(\mathcal{B}^+)$ is at most $k \cdot |\bar{x} - \bar{x}^+|$. On the other hand, with the addition of x_{k+1}, AE increases by at least $|x_{k+1} - \bar{x}^+|$ because this is a new item. Since $x_{k+1} = (k+1)\bar{x}^+ - (x_1 + \cdots + x_k) = (k+1)\bar{x}^+ - k\bar{x}$, then $|x_{k+1} - \bar{x}^+| = |k \cdot (\bar{x} - \bar{x}^+)| = k \cdot |\bar{x} - \bar{x}^+|$. Hence, $\mathrm{AE}(\mathcal{B}^+) - \mathrm{AE}(\mathcal{B}) \geq k \cdot |\bar{x} - \bar{x}^+| - k \cdot |\bar{x} - \bar{x}^+| = 0$ always holds. Therefore, since $\mathcal{B}_{p_i}$ always has more elements than $\mathcal{B}_{p_{i+1}}$, $\mathrm{AE}(\mathcal{B}_{p_i}) \geq \mathrm{AE}(\mathcal{B}_{p_{i+1}})$. □

Considering $\mathrm{BAE}(\mathcal{B})$, there exists a natural number m $(1 \leq m \leq k)$ where if $i < m$, $\mathrm{BAE}(\mathcal{B}_{p_i})) \geq \mathrm{BAE}(\mathcal{B}_{p_{i+1}}) + \delta \geq \theta + 2 - \delta$, if $i = m$, $\theta + 2 \geq \mathrm{BAE}(\mathcal{B}_{p_i}) \geq$

$\theta + 2 - \delta$, and if $m < i$, $\mathrm{BAE}(\mathcal{B}_{p_i}) = \theta + 2 - \delta$. Therefore, using Eqs.(7.16, 7.17),

$$\begin{aligned}(*2) &\leq \frac{2}{\lambda} + \sum_{i=1}^{m-1} \frac{2}{\lambda} \exp\left(\frac{\theta - \mathrm{BAE}(\mathcal{B}_{p_i}) + 2}{\lambda}\right) + \frac{2}{\lambda} \\ &\leq \frac{4}{\lambda} + \frac{2}{\lambda} \cdot \frac{1}{1 - \exp\left(-\frac{\delta}{\lambda}\right)} \\ &= \frac{2}{\lambda} \cdot \frac{3\exp\left(\frac{\delta}{\lambda}\right) - 2}{\exp\left(\frac{\delta}{\lambda}\right) - 1}.\end{aligned} \tag{7.18}$$

Thus, to make $(*2)$ satisfy $\gamma\epsilon_r$-DP, $\frac{2}{\lambda} \cdot \frac{3\exp\left(\frac{\delta}{\lambda}\right)-2}{\exp\left(\frac{\delta}{\lambda}\right)-1} \leq \gamma\epsilon_r$ should holds. Since the λ and δ that satisfy these conditions are not uniquely determined, these values are determined by giving $\exp(\delta/\lambda)$ as a hyperparameter α. Then, we can always calculate $\lambda = (\frac{3\alpha-2}{\alpha-1}) \cdot (\frac{2}{\gamma\epsilon_r})$ and $\delta = \lambda \log \alpha$, in turn, which satisfies $(*2) \leq \gamma\epsilon_r$. α is valid for $\alpha > 1$. If α is extremely close to 1, λ diverges and *random convergence* is too inaccurate. As α increases, λ decreases, but δ increases. Thus λ and δ are trade-offs, and independently of the dataset, there exists a point at which both values are reasonably small. Around $\alpha = 1.4 \sim 1.8$ works well empirically.

Finally, together with $(*1)$, the recursive bisection by random converge and random cut satisfies ϵ_r-DP. In addition, the perturbation consumes ϵ_p for each block to add Laplace noise, so together with this, HDPView satisfies $\epsilon_p + \epsilon_r = \epsilon_b$-DP.

Error Analysis

When a p-view created by HDPView publishes a counting query answer, we can dynamically estimate an upper bound distribution of the error included in the noisy answer. The upper bound of the error can be computed from the number of blocks used to answer the query and the distribution of the perturbation. Note that this can be computed without consuming any extra privacy budget because, as shown in A.5, in addition to the count values, block partitioning results are released in a DP manner.

As a count query on the p-view is processed as Eq. (7.9), the answer consists of the sum of the query results for each block, and from Eq. (7.3), each block contains two types of errors: AE and PE. Let the error of a counting query q be $Error(q, \mathcal{X}, \mathcal{X}') := ||q(\mathcal{X}) - q(\mathcal{X}')||_1$ where $\mathcal{X}$ and $\mathcal{X}'$ are the original and noisy data, respectively, and we define the error by the L1-norm. First, since

the AE depends on the concrete count values of each block involved in each query condition, we characterise the block distribution by defining an ξ-uniformly scattered block.

Definition 13 (ξ-uniformly scattered)**.** *A block $\mathcal{B}$ is ξ-uniformly scattered if for any subblock $\mathcal{B}' \subset \mathcal{B}$,*

$$AE(\mathcal{B}')/|\mathcal{B}'| \leq \xi \cdot AE(\mathcal{B})/|\mathcal{B}|. \tag{7.19}$$

While ξ depends on the actual data, it is expected to decrease with each step by random cut.

Then, we have the following theorem for the error.

Theorem 10. *If for all i, block $\mathcal{B}_i$ is ξ_i-uniformly scattered, any μ satisfying $0 < \mu < 1$, and any t satisfying $|t| < \epsilon_p$ and $|t| < \frac{1}{\lambda}$, the error of a counting query satisfies $Error(q, \mathcal{X}, \mathcal{X}') \geq \Theta_{min}(\mu)$ and $Error(q, \mathcal{X}, \mathcal{X}') \leq \Theta_{max}(\mu)$ with probability of at least $1 - \mu$, respectively, with*

$$\begin{aligned}
\Theta_{min}(\mu) &= \frac{1}{t}\left(\log\mu + \sum_{i=1,\ldots,m} \log\left(1 - (\frac{w_i}{\epsilon_p})^2 t^2\right)\right) \\
\Theta_{max}(\mu) &= \sum_{i=1,\ldots,m} \xi_i w_i (k_i\delta + \theta) \\
&\quad - \frac{1}{t}\left(\log\mu + \sum_{i=1,\ldots,m} \log\left(1 - (\frac{w_i}{\epsilon_p})^2 t^2\right) + \log\left(1 - (\xi_i w_i \lambda)^2 t^2\right)\right)
\end{aligned} \tag{7.20}$$

where $w_i = \frac{|\mathcal{B}_i \cap c_q|}{|\mathcal{B}_i|}$, k_i is depth of $\mathcal{B}_i$ that can be public information.

Proof. The errors included in $Error(q, \mathcal{X}, \mathcal{X}')$ are PEs and AEs. Both of them follow independent probability distributions for each block, and we first show the PE. For each $\mathcal{B}_i$, perturbation noise is uniformly divided inside $\mathcal{B}_i$. Hence, the total PE in the query q is represented by $\sum_{i=1,\ldots,m} w_i * PE(\mathcal{B}_i)$ where $w_i = \frac{|\mathcal{B}_i \cap c_q|}{|\mathcal{B}_i|}$ and $PE(\mathcal{B}_i)$ is Laplace random variable following $Lap(\frac{1}{\epsilon_p})$.

Then, we consider the AE. From random converge, given a $\mathcal{B}_i$, then $BAE(\mathcal{B}_i) + Lap(\lambda) \leq \theta$ holds. Considering $\mathrm{BAE}(\mathcal{B}) = \max(\theta + 2 - \delta, \mathrm{AE}(\mathcal{B}) - k\delta)$, when $\theta + 2 - \delta \leq \mathrm{AE}(\mathcal{B}_i) - k_i\delta$,

$$\mathrm{AE}(\mathcal{B}_i) \leq Lap(\lambda) + k_i\delta + \theta. \tag{7.21}$$

And when $\theta + 2 - \delta > \mathrm{AE}(\mathcal{B}_i) - k_i\delta$,

$$\mathrm{AE}(\mathcal{B}_i) - k_i\delta < \theta + 2 - \delta \leq Lap(\lambda) + \theta \tag{7.22}$$

Thus, the upper bound of $\mathrm{AE}(\mathcal{B}_i)$ is distributed under $Lap(\lambda) + k_i\delta + \theta$. In other words, AE cannot be observed directly, but the upper bound distribution is bounded by the Laplace distribution. Also note that the AE satisfies $\mathrm{AE}(\mathcal{B}_i) \geq 0$.

Therefore, for the error lower bound, we only need to consider the m PEs, $\sum_{i=1,\ldots,m} w_i * PE(\mathcal{B}_i)$. $PE(\mathcal{B}_i)$ is independent random variable, respectively. We apply Chernoff bound to the sum, for any a and t,

$$\Pr\left[Error(q, \mathcal{X}, \mathcal{X}') \leq a\right] \leq e^{(ta)} \prod_{i=1,\ldots,m} E[e^{(-tw_i PE(\mathcal{B}_i))}], \tag{7.23}$$

where $|t| < \epsilon_p$ is required for existence of the moment generating function. By using $PE(\mathcal{B}_i)$ follows $Lap(\frac{1}{\epsilon_p})$, we can derive

$$\Pr\left[Error(q, \mathcal{X}, \mathcal{X}') \leq \frac{1}{t}\left(\log\mu + \sum_{i=1,\ldots,m} \log\left(1 - (\frac{w_i}{\epsilon_p})^2 t^2\right)\right)\right] \leq \mu. \tag{7.24}$$

On the other hand, for the error upper bound, we need to consider AEs as well. Hence, we apply Chernoff bound to the sum of $2m$ independent random variables following each Laplace distribution. Considering the upper bound distribution of $AE(\mathcal{B}_i)$ has $w_i(k_i\delta + \theta)$ for the mean and λ for the variance, let $\bar{AE}(\mathcal{B}_i)$ be a Laplace random variable whose mean and variance are 0 and λ, respectively, then we have

$$\begin{aligned}\Pr\Big[Error(q, \mathcal{X}, \mathcal{X}') - \sum_{i=1,\ldots,m} \xi_i w_i(k_i\delta + \theta) \geq a\Big] \\ \leq e^{(-ta)} \prod_{i=1,\ldots,m} E[e^{(tw_i PE(\mathcal{B}_i))}] E[e^{(tw_i\xi_i \bar{AE}(\mathcal{B}_i))}],\end{aligned} \tag{7.25}$$

where since block $\mathcal{B}_i$ is ξ_i-uniformly scattered, AE included in the query q and block $\mathcal{B}_i$ is at most $\xi_i w_i AE(\mathcal{B}_i)$. Lastly, for any t, where $|t| < \epsilon_p$ and $|t| < \frac{1}{\lambda}$, from the inequality, we can derive as follows:

$$\begin{aligned}&\Pr\Bigg[Error(q, \mathcal{X}, \mathcal{X}') \geq \sum_{i=1,\ldots,m} \xi_i w_i(k_i\delta + \theta) \\ &\quad - \frac{1}{t}\left(\log\mu + \sum_{i=1,\ldots,m} \log\left(1 - (\frac{w_i}{\epsilon_p})^2 t^2\right) + \log\left(1 - (\xi_i w_i \lambda)^2 t^2\right)\right)\Bigg] \\ &\leq \mu.\end{aligned}$$

This completes the proof. □

Table A.3: Datasets.

Dataset	#Record	#Column (categorical)	#Domain	Variance
`Adult`	48842	15 (9)	9×10^{19}	0.0360
`Small-adult`	48842	4 (2)	3×10^{5}	0.0237
`Numerical-adult`	48842	7 (1)	2×10^{11}	0.0200
`Traffic`	48204	8 (2)	1×10^{14}	0.0484
`Bitcoin`	500000	9 (1)	4×10^{12}	0.0379
`Electricity`	45312	8 (1)	1×10^{14}	0.0407
`Phoneme`	5404	6 (1)	2×10^{6}	0.0304
`Jm1`	10885	22 (1)	2×10^{21}	0.0027

Importantly, this can be dynamically computed for any counting queries, helping the analyst to perform a reliable exploration.

Similarly, since the HDMM [113] optimizes budget allocations for counting queries by the MM, we can statically calculate the error distributions for each query. However, this is workload-dependent. In data exploration, we consider predefined workload is strong assumption to be avoided.

A.6 Evaluation

In this section, we report the results of the experimental evaluation of HDPView. To evaluate our proposed method, we design experiments to answer the following questions:

- How effectively can the constructed p-views be used in data exploration via various range counting queries?
- How space-efficiently can the constructed p-views represent high-dimensional count tensors?

We shows the effectiveness of HDPView via range counting queries in section A.6, and section A.6 reports the space efficiency.

Experimental Setup

We describe the experimental setups. In the following experiments, we run 10 trials with HDPView and the competitors and report their averages to remove bias. Throughout the experiments, the hyperparameters of HDPView are fixed as $(\epsilon_r/\epsilon_b, \alpha, \beta, \gamma) = (0.9, 1.6, 1.2, 0.9)$. Please see Section A.6 for insights on how to determine the hyperparameters. We provide observations and insights into all the hyperparameters of HDPView later.

Datasets. We use several multidimensional datasets commonly used in the literature, as shown in Table A.3. `Adult`[b] includes 6 numerical and 9 categorical attributes. We prepare `Small-adult` by extracting 4 attributes (age, workclass, race, and capital-gain) from `Adult`. Additionally, we form `Numerical-adult` by extracting only numerical attributes and a label. `Traffic`[c] is a traffic volume dataset. `Bitcoin`[d] is a Bitcoin transaction graph dataset. `Electricity`[e] is a dataset on changes in electricity prices. `Phoneme`[f] is a dataset for distinguishing between nasal and oral sounds. `Jm1`[g] is a dataset of static source code analysis data for detecting defects with 22 attributes. HDPView and most competitors require the binning of all numerical attribute values for each dataset. Basically, we set the number of bins to 100 or 10 when the attribute is a real number. We consider that the number of bins should be determined by the level of granularity that analysts want to explore, regardless of the distribution of the data. For categorical columns, we simply apply ordinal encoding. In Table A.3, #Domain shows the total domain sizes after binning. Variance is the mean of the variance for each dimension of the binned and normalized dataset and gives an indication of how scattered the data is.

Implementations of competitors. We compare our proposed method HDPView with Identity [42], Privtree [114], HDMM [113], PrivBayes [242], and DAWA partitioning mechanism [248]. For these methods, we perform the following pre- and postprocessing steps. For Identity, we estimate errors following [113], employing implicit matrix representations and workload-based estimation, because it is infeasible to add noises on a high-dimensional count tensor because

[b] `http://archive.ics.uci.edu/ml/datasets/Adult`
[c] `http://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume`
[d] `https://archive.ics.uci.edu/ml/datasets/BitcoinHeistRansomwareAddressDataset`
[e] `https://www.openml.org/d/151`
[f] `https://www.openml.org/d/1489`
[g] `https://www.openml.org/d/1053`

of the huge space. For Privtree, as described in [114], we set the threshold to 0 and allocate half of the privacy budget to tree construction and half to perturbation. Using the same method as HDPView, the blocks obtained by Privtree are used as the p-view. For the HDMM, we utilize p-Identity strategies as a template. DAWA's partitioning mechanism can be applied to multidimensional data by flattening data into a 1D vector. However, when the domain size becomes large, the optimization algorithm based on the v-optimal histogram for the count vector cannot be applied due to the computational complexity. Therefore, we apply DAWA to `Small-adult` and `Phoneme` because their domain sizes are relatively small. We perform only DAWA partitioning without workload optimization to compare the partitioning capability without a given workload to evaluate workload-independent p-view generation. For fairness, PrivBayes is trained on raw data[h]. PrivBayes, in counting queries, samples the exact number of original data points; therefore, it may consume extra privacy budget.

Workloads. We prepare different types of workloads. *k-way All Marginal* is all marginal counting queries using all combinations of k attributes. *k-way All Range* is the range version of the marginal query. *Prefix-kD* is a prefix query using all combinations of k attributes. *Random-kD Range Query* is a range query for arbitrary k attributes and we randomly generate 3000 queries for a single workload.

Reproducibility. The experimental code is publicly available [i].

Effectiveness

We evaluate how effective p-views constructed by HDPView are in data exploration by issuing various range counting queries. We utilize the above-mentioned workloads.

Evaluation metrics. We evaluate HDPView and other mechanisms by measuring the RMSE for all counting queries. Formally, given the count tensor $\mathcal{X}$, randomized view $\mathcal{X}'$ and workload $\mathbf{W}$, the RMSE is defined as: $\text{RMSE} = \sqrt{\frac{1}{|\mathbf{W}|}\sum_{q\in\mathbf{W}}(q(\mathcal{X}) - q(\mathcal{X}'))^2}$. This metric is useful for showing the utility of the p-view. It corresponds to the objective function optimized by MM families [249, 113], where given a workload matrix $\boldsymbol{W}$ and a query strategy $\boldsymbol{A}$, which is the optimized query set to answer the workload, the expected error of the

[h] PrivBayes shows worse performances with binned data in our prestudy.

[i] `https://github.com/FumiyukiKato/HDPView`

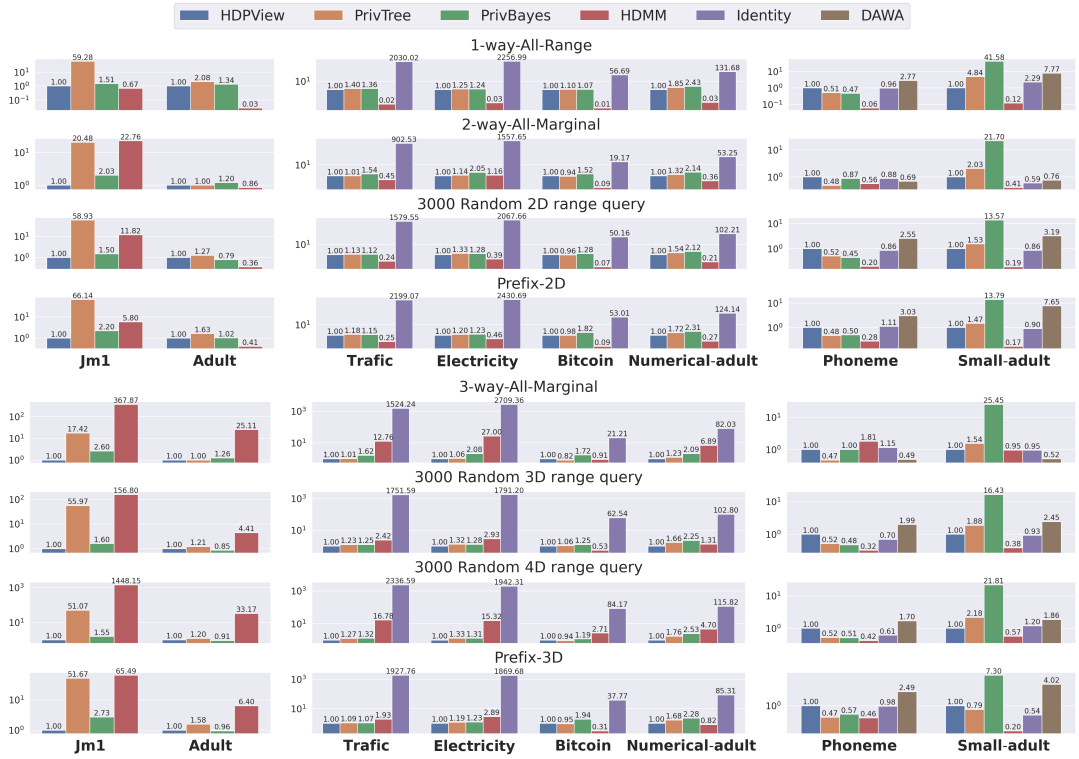
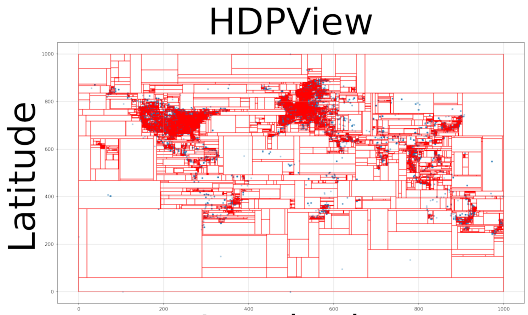


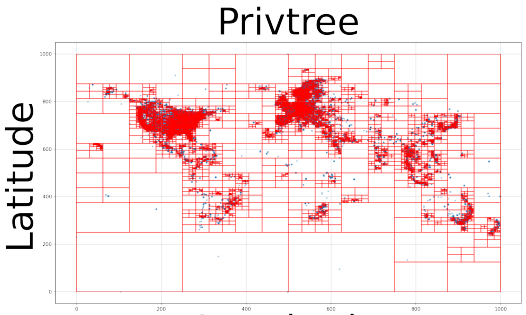
Figure A.3: Relative RMSEs against HDPView on all the datasets and workloads: HDPView shows small errors for a wide variety of high-dimensional range counting queries.

workloads is $\frac{2}{\epsilon^2}||\boldsymbol{A}||_1^2||\boldsymbol{W}\boldsymbol{A}^+||_F^2 = \text{RMSE}^2$. Thus, we can compare the measured errors with this optimized estimated errors. We also report the relative RMSE against HDPView for comparison. To evaluate the average performance for a wide range of queries and datasets, we compute the average over all workloads and all datasets. We refer to this metric as *average relative RMSE* (ARR).

High quality on average. Figure A.3 shows the relative RMSEs for all datasets and workloads and algorithms with privacy budget ϵ=1.0. The relative RMSE (log-scale) is plotted on the vertical axis and the dataset on the horizontal axis where high-dimensional datasets (`Jm1` and `Adult`) are on the left, medium-dimensional datasets (`Traffic`, `Electricity`, `Bitcoin` and `Numerical-adult`) are in the middle, and low-dimensional datasets (`Small-adult` and `Phoneme`) are on the right. The errors with Identity for high-dimensional data are too large and are omitted for appearance. As a whole, HDPView works well. In Section A.1, Table A.2 shows the relative RMSE averaged over all workloads and all datasets in



#Blocks	15669
RMSE	657.37



#Blocks	19475
RMSE	315.79

Figure A.4: Examples of HDPView (left) and Privtree (right) on 2D dataset (`Gowalla`): HDPView has fewer blocks, leading to noisier results than Privtree for very low-dimensional data. Also, HDPView provides flexible block partitioning.

Figure A.3, and HDPView achieves the lowest error on average. In data exploration, we want to run a variety of queries, so the average accuracy is important. We believe HDPView has such a desirable property. A detailed comparisons with the competitors are explained in the following paragraphs.

Comparison with Identity, HDMM and DAWA. Identity, which is the most basic Baseline, and HDMM, which performs workload optimization, cause more errors for high-dimensional datasets than HDPView. For Identity, the reason is that the accumulation of noise increases as the number of domains increases. HDPView avoids the noise accumulation by grouping domains into large blocks. The results of HDMM show that the increasing dimension of the dataset and the dimension of the query can increase the error. This is because the matrix representing the counting queries to which the matrix mechanism is applied becomes complicated, making it hard to find efficient budget allocations. This is why the accuracy of the 3- or 4-dimensional queries for `Jm1` and `Adult` is poor with HDMM. In particular, the HDMM's sensitivity to dimensionality increases can also be seen in Figure A.7. DAWA's partitioning leads more errors than the HDPView and Privtree. When applied to multi-dimensional data, DAWA finds the optimal partitioning on a domain mapped in one-dimension, while HDPView and Privtree finds more effective multi-dimensional data partitioning.

Comparison with Privtree. Overall, HDPView outperforms Privtree's ac-

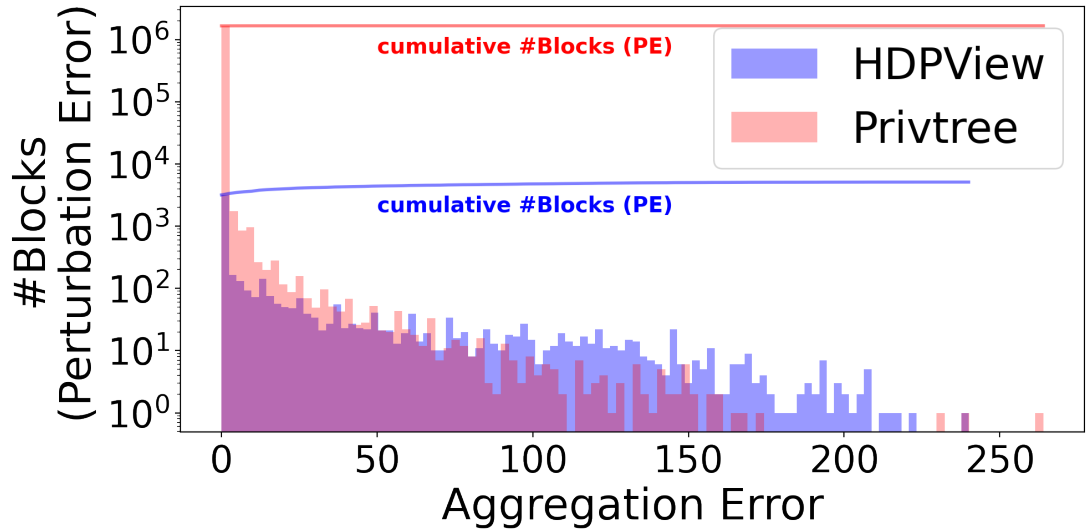


Figure A.5: Number of blocks (log-scale) with various AEs for high-dimensional dataset (i.e., `Adult`) for HDPView and Privtree. HDPView has slightly larger AE blocks, but Privtree has a much more number of blocks i.e., much larger PEs.

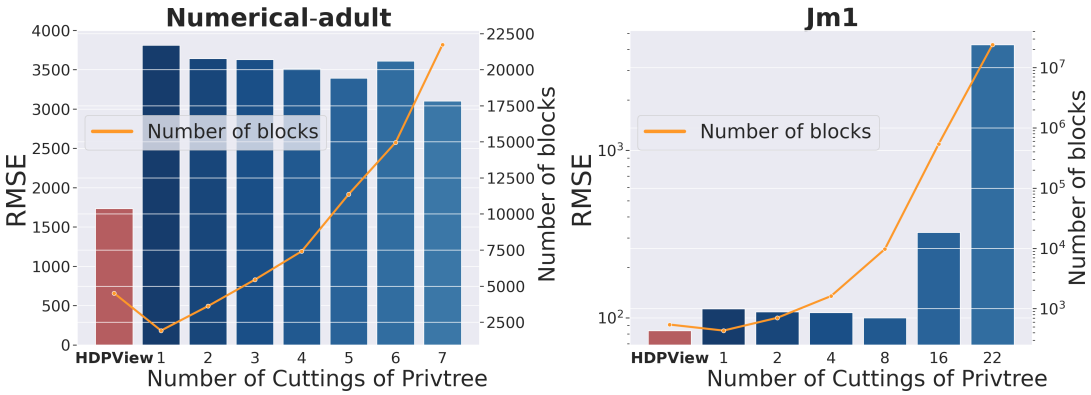


Figure A.6: HDPView is more effective than Privtree even with controlled number of cuttings on `Numerical-adult` (left) and `Jm1` (log-scale) (right).

curacy mainly for mid- to high-dimensional datasets. In particular, we can see Privtree's performance drops drastically in high-dimensionality (i.e., `Jm1`). Privtree achieves higher accuracy than HDPView for `Phoneme`. This is likely because Privtree's strategy, which prioritize finer splitting, are sufficient for the small domain size rather than HDPView's heuristic algorithm. Even if the blocks is too fine, the accumulation of PEs is not so large in low-dimensionality, and AEs become smaller, which results in an accurate p-view. The reason why

HDPView is better for `Small-adult` despite the low-dimensionality may be that the sizes of the cardinality of attributes are uneven (`Small-adult`: {74, 9, 5, 100}, `Phoneme`: {10, 10, 10, 10, 10, 10, 2}), which may make Privtree's fixed cutting strategy ineffective. To see the very low-dimensional case, Figure A.4 shows the block partitioning for the 2D data with a popular `Gowalla`[j] check-in dataset. The table below shows the number of blocks and the RMSE for the 3000 Random 2D range query. HDPView yields fewer blocks and Privtree generates a less noisy p-view for the above-mentioned reason. The figure also confirms that HDPView performs a flexible shape of block partitioning.

On the other hand, for high-dimensional dataset, this property can be avenged. In Privtree, a single cutting always generates 2^d new blocks, which are too fine, resulting in very large PEs even though the AEs are smaller. Figure A.5 shows the distribution of AEs for blocks on `Adult` for HDPView and Privtree. HDPView has slightly larger AE blocks, but Privtree has a large number of blocks and cause larger PEs. An extreme case is `Jm1` in which Privtree causes large errors. This is probably because `Jm1` actually requires fewer blocks since the distribution is highly concentrated (c.f., Table A.3). Figure A.8 shows that the number of blocks of generated p-view by HDPView and Privtree. For `Jm1`, HDPView generates very small number of blocks while Privtree does not. We can confirm that HDPView avoids unnecessary splitting via *random cut* and suppresses the increase in PEs which causes in Privtree. This would be noticeable for datasets with concentrated distributions, where the required number of blocks is essentially small. In summary, Privtree's partitioning strategy contributes to smaller AEs, but leads to an excessive increase in PEs for high-dimensional data.

Figure A.6 shows the results of reducing the number of cut attributes in Privtree and adjusting the number of blocks in p-view on `Numerical-adult` and `Jm1`. If the number of cut attributes is smaller than the dimension d, we choose target attributes in a round-robin way (Appendix of [114]). In the case of `Numerical-adult`, the error basically decreases as the number of cut attributes is increased, similar to the observation in Appendix of [114]. However, for high-dimensional data such as `Jm1`, the error increases rapidly as the number of cut attributes increases to some extent. This is consistent with the earlier observation that influence of PEs increases. Also, in any cases, the error of HDPView is smaller, indicating that HDPView not only has a smaller number of blocks, but

[j] `http://snap.stanford.edu/data/loc-Gowalla.html`

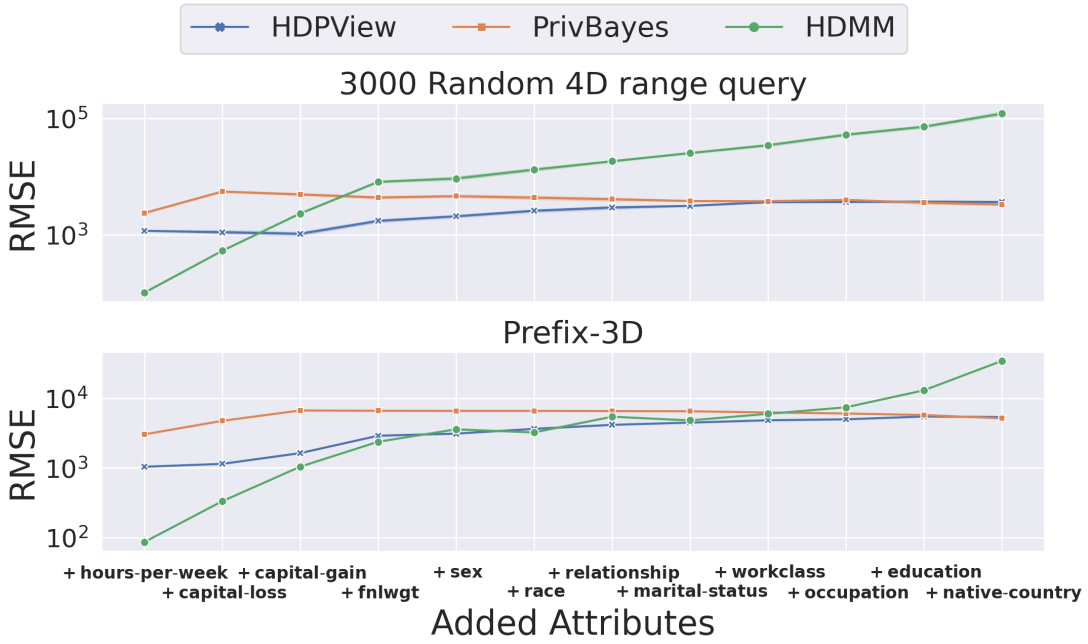


Figure A.7: Changes in the performance when adding attributes to `Adult` one by one in HDPView, PrivBayes, and HDMM.

also performs effective block partitioning compared to Privtree on these datasets.

Comparison with PrivBayes. We do not consider PrivBayes a direct competitor because it is a generative model approach that does not provide any analytical reliability as described in Section A.2. However, PrivBayes is a state-of-the-art specialized for publishing differentially private marginal queries; therefore, we compared the accuracy to demonstrate the performance of HDPView. As shown in Figure A.3, HDPView is a little more accurate than PrivBayes in many cases. However, in `Adult`, PrivBayes slightly outperforms HDPView. Because PrivBayes uses Bayesian network to learn the data distribution, it can fit well even to high-dimensional data as long as the distribution of the data is easily modelable. In HDPView, with larger dimensionality, the PEs grow slightly because the total number of blocks increases. The AEs also grow since more times of random converge result in larger errors. Thus, the total error is at least expected to increase, and the larger dimensionality may work to the advantage of PrivBayes. Still, HDPView is advantageous, especially for concentrated data such as `Jm1`.

We consider the reason why on `Numerical-adult`, which has a smaller di-

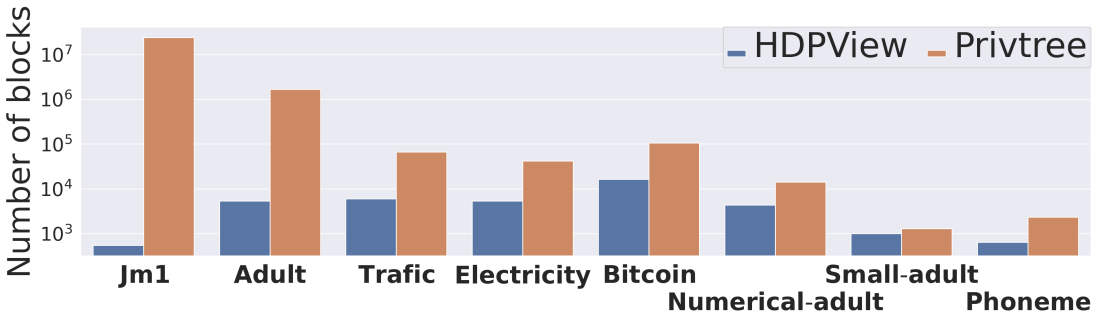


Figure A.8: The number of blocks generated by HDPView is much lower than that generated by Privtree.

mensionality than `Adult`, PrivBayes is less accurate than HDPView is because the effective attributes for capturing the accurate marginal distributions with Bayesian network are removed. We can see the same behavior for `Small-adult`. The following experimental results can support this. Figure A.7 describes the changes in the RMSE with attributes added to `Adult` one by one in two workloads, where the added attributes are shown on the horizontal axis. Initially, HDPView is more accurate than PrivBayes. As attributes are added, HDPView is basically robust with increasing dimensionality, but the error increases slightly. On the other hand, interestingly, the error in PrivBayes becomes slightly smaller.

Lastly, considering HDPView is better in `Numerical-adult` and worse in `Adult`, one of the advantages of PrivBayes may be due to the increase in categorical attributes. Because HDPView bisects the ordered domain space, it may be hard to effectively divide categorical attributes, which possibly worsens the accuracy in HDPView.

Space Efficiency

Our proposed p-view stores each block in a single record. This method avoids redundancy in recording all cells that belong to the same block. The p-view consists of blocks and values, and basically, the space complexity follows the number of blocks. Figure A.8 shows a comparison between the numbers of blocks of HDPView and Privtree. While the accuracy of the counting queries of HDPView is higher than that of Privtree, the number of blocks generated by HDPView is much lower than that of Privtree, indicating that the strategy of HDPView

Table A.4: HDPVIEW's p-view is space efficient (up to $10^{13}\times$).

Dataset	Identity-based	HDPVIEW
`Adult`	30.99 EB	3.61 MB
`Bitcoin`	1.27 TB	6.77 MB
`Electricity`	1.11 TB	2.19 MB
`Phoneme`	781.34 KB	273.59 KB

avoids unnecessary splitting. In particular, on `Jm1`, HDPVIEW is 4×10^4 more efficient than Privtree. Table A.4 shows the size of the randomized views, Identity-based noisy count vector (not p-view) and p-view generated by HDPVIEW at ϵ=1.0. Since HDPVIEW constructs the p-view by a compact representation, it results in up to 10^{13} times smaller space on `Adult`.

Analysis for Hyperparameters

We provide an explanation of the hyperparameters of HDPVIEW. As shown in Algorithm 16, HDPVIEW requires four main hyperparameters, ϵ_r/ϵ_b, α, β and γ. We mentioned in Section A.6 that we fix the hyperparameters as $(\epsilon_r/\epsilon_b, \alpha, \beta, \gamma) = (0.9, 1.6, 1.2, 0.9)$ in the experiments. Here, we provide some insights into each hyperparameter from observations of experimental results on a real-world dataset `Small-adult` varying each hyperparameter.

Figures A.9 A.10 A.11 A.13 show the RMSEs for *3000 random 2D range query* on `Small-adult` dataset when only one of the hyperparameters varies and others are fixed as the above-mentioned default. From this result, we obtain the following insights:

Ratio, ϵ_r/ϵ_b. The best accuracy is achieved when the Ratio is approximately 0.7˜0.9 as shown in Figure A.9. In HDPVIEW, seemingly, the effect of aggregation error is larger than perturbation error, therefore we try to allocate a budget to the cutting side so that the aggregation error is smaller.

Gamma, γ. As shown in Figure A.10, it was confirmed that prioritizing the*random converge* to accurately determine AEs improves accuracy rather than *random cut*. However, if no budget is allocated for *random cut*, the error increases, i.e., a completely random cutting strategy lose accuracy compared to appropriate our proposed *random cut*. Therefore, 0.9 is reasonable. However, the random cut

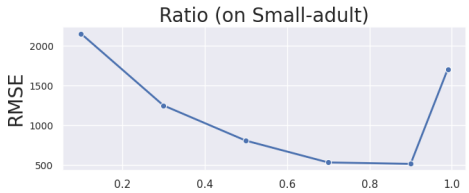


Figure A.9: Effects of HDPView's hyperparameter ϵ_r/ϵ_b (Ratio) on `Small-adult` dataset.

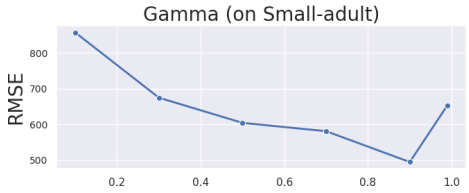


Figure A.10: γ on `Small-adult`.

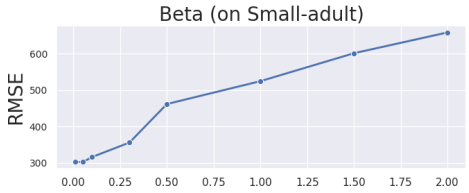


Figure A.11: β on `Small-adult`.

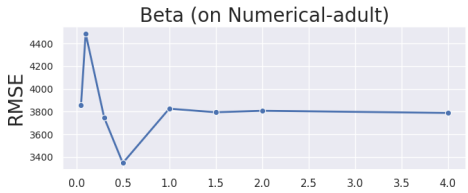


Figure A.12: β on `Numerical-adult` dataset: Optimal β depends on dataset and our default parameter $\beta = 1.2$ is somewhat conservative.

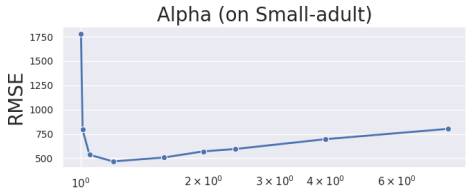


Figure A.13: α on `Small-adult`.

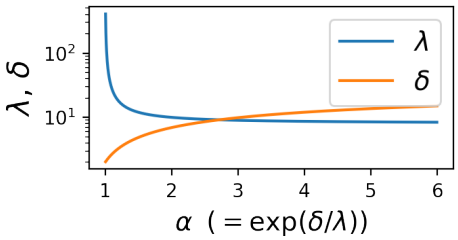


Figure A.14: λ and δ on various α when $\gamma\epsilon_r = 1.0$.

may be less less significant due to the conservative setting of the β shown below.

Beta, β. The β is somewhat conservatively determined. We choose $\beta = 1.2$ because, when $\beta = 1.2$, the maximum depth of HDPView 's bisection rarely reaches κ on various datasets. As a rough guideline, if the total number of domains in a given dataset is $\tilde{n}$, all blocks can be split at $\log_2 \tilde{n}$ times depth, assuming the domains are bisected exactly in the all cutting. Thus $\kappa = 1.2 * \log_2 \tilde{n}$ is deep enough to split all the blocks, allowing for some skewness, and all the cutting point is likely to be selected by an Exponential Mechanism rather than by random one. However, remember the budget for each EM is inversely proportional to κ, depending on the data set, the budget available for EM may be unnecessarily small due to the unnecessarily large setting of κ as is the case with `Small-adult`

(Figure A.11). We also show the `Numerical-adult` result as another example (Figure A.12). The small β do not take full advantage of the random cut. Since determining the optimal β for any dataset is impossible without additional privacy consumption, we conservatively set $\beta = 1.2$ for all dataset in the experiment.

Alpha, α. α, i.e., $\exp(\delta/\lambda)$, is valid for $\alpha > 1$. If α is extremely close to 1, λ diverges and HDPView does not work well because random converge causes large errors. Because $\lambda = (\frac{3\alpha-2}{\alpha-1}) \cdot (\frac{2}{\gamma\epsilon_r})$ and $\delta = \lambda \log \alpha$, as α increases, λ decreases and converges to 3, but δ increases. Thus λ and δ are trade-offs. When δ increases, the bias of BAE increases, which also leads to a worse convergence decision. Figure A.14 plots the size of λ and δ for various α when $\gamma\epsilon_r = 1.0$. As α increases, the δ increases, but the decrease in λ is small, starting from approximately 1.4. Therefore, around $\alpha = 1.4 \sim 1.8$ works well empirically and we use $\alpha = 1.6$ as default value.

A.7 Conclusion

We addressed the following research question: How can we construct a privacy-preserving materialized view to explore the unknown properties of the high-dimensional sensitive data? To practically construct the p-view, we proposed a data-aware segmentation method, HDPView. In the experiments, we confirmed the following desirable properties, (1) Effectiveness: HDPView demonstrated smaller errors for various range counting queries in multidimensional queries. (2) Space efficiency: HDPView generates a compact representation of the p-view. We believe that our method helps us explore sensitive data in the early stages of data mining while preserving data utility and privacy.

B PCT-TEE: Trajectory-based Private Contact Tracing System with Trusted Execution Environment

B.1 Introduction

Since the beginning of 2020, the emergence of COVID-19 has caused a worldwide pandemic. Many governments and companies are developing various measures

and technologies to prevent the spread of the virus [262, 263, 264, 265]. At present, contact tracing is expected to be a powerful countermeasure for controlling the spread of infection. The effectiveness of contact tracing has already been shown by several previous studies [266, 267, 268, 269]. However, conducting effective contact tracing often requires collecting citizens' personal information, such as their locations [270] or telephone numbers [271], which raises ethical issues and serious privacy violations [272]. Therefore, acceptable private contact tracing (PCT) is urgently needed.

Recently, Bluetooth-based PCT has been intensively studied [135, 273, 274, 275, 276]. Decentralized privacy-preserving proximity tracing (DP3T) [135], which is an open protocol for PCT that uses Bluetooth low-energy (BLE) beacons, is already being used in applications developed worldwide. To strongly protect users' privacy, it uses only the contact (proximity) history detected by the BLE beacons. In DP3T, the applications use the Bluetooth signal of a smartphone to broadcast a random ID that does not include sensitive information such as the user's identity or location, and nearby smartphone devices receive and store the data for a limited time. Users who are then discovered to be infected with COVID-19 send a report to the server that includes the random IDs they have generated. Moreover, each user routinely checks to see if the random IDs received from devices they have contacted in the past have been uploaded to the server. Additionally, there are similar methods for adopting decentralized architecture, such as Epione [273], the PACT protocol [274], CEN [275, 277] and Google and Apple specifications [276].

However, Bluetooth-based PCT has several limitations in terms of functionality and flexibility. First, Bluetooth-based PCT detects only *direct contact* (i.e., human-human contact) but cannot detect *indirect contact* (i.e., human-object contact, e.g., when a person visits the same shop shortly after a patient with COVID-19 visited it). The Centers for Disease Control and Prevention (CDC) in the US showed that it is possible for a person to become infected with COVID-19 by touching a surface or object that has the virus on it and then touching their own mouth, nose, or eyes [278] — despite not making direct contact with patients with COVID-19. Moreover, recent studies [279, 280] followed this idea and highlighted the need to trace indirect contact. Second, Bluetooth-based PCT lacks flexibility in terms of determining the rule of "risky contact". Essentially, the rule of risky contact in Bluetooth-based PCT is hard-wired into the Bluetooth device

since risky contact is implicitly defined as two devices being in close proximity to each other's signal ranges. In practice, whether or not risky contact occurs varies with the environmental situation and the nature of the virus. The rules of risky contact in the case of COVID-19 have been updated as our understanding of the virus [281] has improved. For example, in the beginning of the pandemic, professionals believed that transmission took place only through direct human-human contact; however, it was recently argued that airborne transmission should be taken into account [282, 279]. In addition to the nature of the virus, a recent epidemiological study [283] showed the importance of appropriate selection of the proximity detection range (PDR), which also supports the necessity of flexibility in PCT. Moreover, recent reviews have pointed out the current PCT application limitations [284, 285], which are the inability to detect infections that do not involve direct contact and radio signal limitations in contact detection [266, 286].

In this work, we propose secure and efficient *trajectory-based PCT* to enable both direct and indirect contact tracing. By comparing the trajectory data between a user and infected patients, we can check whether or not the user visits "infected locations" *within a certain time period*. The rule of risky contact can be flexibly defined according to the condition of the location and the nature of the virus. The four requirements for trajectory-based PCT are as follows.

1. *Efficiency*: The central server must be able to handle the query throughput.
2. *Security*: A client's trajectory data must be protected from the server and any other clients. However, nothing about the server-side data is disclosed to the client except the query result.
3. *Flexibility*: The rule of risky contact should be simple to change when necessary.
4. *Accuracy*: The server should carefully return accurate results because these results are very sensitive and can significantly affect users.

As shown in Figure B.1, we assume that the health agency (e.g., the government or official healthcare institute) registers the trajectory data of patients confirmed to have COVID-19 (these data are encrypted or released under the consent of the patients) to a server that is untrusted by clients (i.e., queriers). The server receives queries and encrypted personal trajectories from clients and returns a

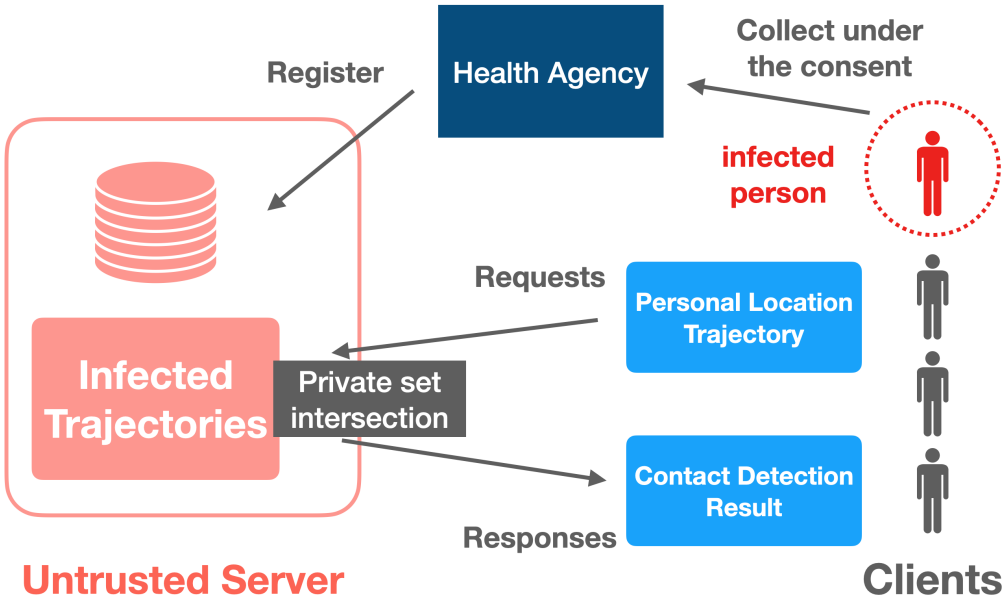


Figure B.1: Trajectory-based PCT overview.

	Functionality	Efficiency	Security	Flexibility	Accuracy
DP3T [135]	-	✓	✓	-	✓
HE-based PSI [287, 64]	-	-	✓	-	✓
TEE-based PSI [288]	-	✓	✓	-	-
MPC-based PCT [267]	✓	-	✓	-	✓
PCT-TEE (mine)	✓	✓	✓	✓	✓

Table B.1: Comparison with existing approaches.

Boolean value indicating whether there is risky contact or not by computing an intersection between server and client trajectories in a private manner.

Although the problem of trajectory-based PCT is similar to the well-studied problem of a private set intersection (PSI), the existing approaches for the PSI cannot satisfy all four of the above-mentioned requirements. A PSI ensures that two (or more) parties can collaboratively calculate the intersection of their private sets without their private data being disclosed to the other party, only the existing information of the intersection or the result. However, the existing PSI techniques, based mostly on cryptographic primitives, cannot achieve all of the above-mentioned requirements. The state-of-the-art cryptography-based PSI ap-

proaches, such as oblivious transfer [287] and homomorphic encryption [64] have limitations in terms of the *efficiency*, and there are still performance problems [289, 267] under medium or large workloads, due mainly to the heavy use of time-consuming cryptographic primitives. Recently, approaches based on secure hardware (such as Intel SGX or ARM TrustZone) have received increasing attention. Secure hardware enables to make trusted execution environment (TEE) [62, 70], which is used to speed up secure computations on an untrusted party. Tamrakar et al. [288] proposed the first efficient TEE-based PSI. It is efficient but does not satisfy the *accuracy* requirement since it introduces unpredictable error because of the use of probabilistic data structures. Moreover, *flexibility* is not considered because their work is based on a general hash function for hash-based dictionary structures. In the method, we use a flexible encoding hash function to satisfy the requirements. Thus, we compare their work with ours in Table B.1, adding [267] which is an MPC-based PCT system using trajectories. Functionality means the capability to detect *indirect contact*.

The contributions in this work are threefold. First, we formulate the problem of trajectory-based PCT. We show that the problem is a generalization of the well-studied private proximity testing [289] and PSI. The formulation is parameterized for both time and space and can be used in general settings. We name this formulation the *spatiotemporal PSI*. Second, we propose PCT-TEE, a TEE-based efficient algorithm for trajectory-based PCT. In addition to satisfying the above-mentioned requirements, a challenge in designing the TEE-based algorithm is the constraint of secure memory (i.e., enclave) on secure hardware. We solve these problems by designing a novel trajectory data encoding method, *TrajectoryHash*, and combining it with *fast succinct trie* [290]. TrajectoryHash and the fast succinct trie enable algorithmic flexibility, more efficient compression, and deterministic and fast search performance for a high-speed PSI in a TEE. Third, we implement the proposed system on Intel SGX and open source the prototype code in GitHub[k]. Our experiments on real-world datasets show that the proposed system is efficient and effective in practical scenarios. Specifically, the proposed encoding and data structure compresses the actual trajectory data to one-sixth the size of the Hashmap (also known as a hash table) with the same performance, and as a result, the total execution time is substantially reduced. Moreover, we show that the proposed system, implemented on a single machine equipped with

[k] `https://github.com/ylab-public/PCT`

SGX, can handle hundreds of queries on tens of millions of records of trajectory data in a few seconds.

Outline. In Section B.2, as preliminaries, we show that certain features of Intel SGX are related to our system design and discuss the TEE-based PSI and conventional cryptography-based PSI performances. In Section B.3, we offer the problem statement and formulate the PCT problem. In Section B.4, we give an overview of our architecture, and in Section B.5, we present the algorithm and trajectory-based data compression. In Section B.6, we show the experimental results and evaluation. In Section B.7, we present related works, including related recent PCT applications and the position of this work. Finally, we provide the conclusions in Section B.8.

B.2 Preliminaries

Characteristics of Intel SGX

As introduced in Section 2.3, Intel SGX [70] is the extended instruction set of Intel x86 processors, which enables the creation of an isolated TEE, called the *enclave*. The enclave resides in the protected memory region, called the Enclave Page Cache (EPC), in which all programs and data can be unencrypted and quickly processed as well as transparently encrypted outside the CPU package by a memory encryption engine using a secret key that only processor hardware can access. In other words, SGX adopts a model that considers the CPU package as a trust boundary and everything outside as untrusted. In this trusted space, accesses from any untrusted software, including the OS/Hypervisor, are prohibited by the CPU, protecting the confidentiality and integrity of the program and data inside the enclave. Therefore, programs using SGX must use two types of instructions called *OCALL/ECALL* to invoke functions across trust boundaries under strict control. These instructions often require too many clock cycles [291], and does uploading data to the enclave. This observation is important to improve the system performance.

Memory size limitation. A challenge in designing algorithms for Intel SGX is the size constraint of the EPC. The maximum size of the EPC is limited to 128 MB, including 32 MB of metadata for secure management (or 256 MB including 64 MB of metadata in the recent Intel high-end processor [292] in 2021). This limitation may be gradually improved but will continue to be a problem regarding

hardware and memory-securing performance. Assume that memory is allocated beyond this memory size constraint. In this case, SGX with Linux allows paging with special encryption. However, many studies have shown that the performance is greatly degraded by severe overhead [293, 294, 134], which is derived from a requirement to preserve confidentiality and integrity even outside the enclave. Therefore, it is necessary to design an efficient algorithm that works within SGX. This is still an important problem, and Kockan et al. [295] presented a method to overcome the severe memory limitation of TEEs for genomic data analysis.

Private Set Intersection

The PSI is a well-studied and important problem. The PSI refers to a setting where multiple parties each hold a set of private sets and wish to learn the intersection of their sets without revealing any information except for the intersection itself. The existing main approach is to use cryptographic primitives, which are summarized as follows. We can classify the conventional approaches into two categories; methodology and security model.

Regarding the former, first, there are methods based on the commutative properties of the Diffie–Hellman (DH) key exchange [296]. They require computing the polynomial interpolation, which requires a high computational cost. Huang et al. [297] described a garbled circuit-based approach. Their proposed SCS circuit family improved the efficiency at that time. This approach is similar to the secure-hardware-based approach described later in terms of leveraging a general-purpose secure computation. Oblivious transfer (OT) [298, 287, 299] is one of the most promising approaches. While it is generally used for semi-honest adversaries, [287] extended the OT method to a malicious adversary using the dual execution technique [300]. Homomorphic encryption (HE) [64, 301] is suitable for an unbalanced setting where the server-side data are large and the client-side data are small, because it can replace the oblivious pseudorandom function in the OT-based approach with leveled fully HE and substantially reduce the amount of data to be transmitted. Last, there is a method extended from private information retrieval [302]. Thus, many improvements have been proposed based on these extensions; however, there is still no method to achieve practical efficiency on large scale data in terms of the execution time and communication bandwidth.

Regarding the latter category, there are *semi-honest* [303] or *malicious* adversaries. Roughly speaking, a semi-honest adversary is an attacker who tries to

	bandwidth	**computation**	**requirements**	**server security**
TEE	$\mathcal{O}(n)$	$\mathcal{O}(n)$	attestation hardware	malicious
OT [287]	$\mathcal{O}(N)$	$\mathcal{O}(N \log N)$	nothing	malicious
HE [64]	$\mathcal{O}(n \log N)$	$\mathcal{O}(N)$	nothing	semi-honest

Table B.2: PSI comparison: cryptography-based(OT, HE) v.s. TEE-based: TEE needs to special hardware, but it has advantages in efficiency and security.

infer secret information from the information he obtains, while following correct protocols and not crafting send and receive data, while a malicious adversary is an attacker who crafts send and receive data and executes the protocol as many times as possible to extract secret information. Generally, the malicious client setting requires a more secure standard and higher costs. Which model we should secure depends on the applications and situations, but in this scenario, we consider a malicious adversary because it is reasonable to consider that an untrusted server can control and access any computation on the server. The TEE-based approach can achieve malicious security [304].

We consider the secure-hardware-based approach to be the better option. Regarding the methodology, we do not have to use the above-mentioned cryptographic primitives. Using Intel SGX, platform verification and transparent memory encryption by the hardware occurs so quickly and thoroughly that a highly efficient PSI can be achieved. The difference in efficiency is significant, being especially impactful on the choice. Additionally, the TEE provides a refined security model for a malicious adversary. [304] shows that no operation or inputs expose information regarding the inner state or data of the TEE. All we need to consider is privacy leakage from outside the TEE and software implementation bugs.

We present a deeper comparison between the existing cryptography-based PSI and the secure-hardware-based PSI in terms of the efficiency. We recognize the state-of-the-art approaches as [287] in terms of the balanced data size setting in the server and client and [64] in terms of the unbalanced case. The latter assumes a semi-honest server and is faster than others in the malicious server model. Table B.2 shows a comparison of the properties between them and secure

hardware (Intel SGX). It includes a relatively rough estimate of the asymptotic bandwidth and computational costs at every PSI execution. We denote n, N as the client data size and server data size, respectively, assuming $n << N$. Asymptotic comparisons are acceptable for one purpose because of the large impact of different coefficients. However, with secure hardware, both communication and computing costs are dramatically more efficient, as they are proportional only to the client data size (Table B.2). On the other hand, the secure-TEE-based approach requires RA in advance and a hardware with special functionality on the server side. Cryptography-based methods do not need any special devices; they consist solely of algorithms. Rindal et al. [287] reduce communication cost to (N) from the naive (N^2) cost by using a variant of Cuckoo hashing in an OT-based method. In [64], the communication cost is efficiently reduced to $\mathcal{O}(n \log N)$, and the server computational cost is $\mathcal{O}(n)$ homomorphic evaluations on large circuits of size $\mathcal{O}(N/n)$. These facts show that the secure-hardware-based method demands an extra cost such as special hardware, but is a better choice for large-scale deployment because of the significant efficiency.

B.3 Problem Formulation

We first introduce the trajectory-based PCT scenario, and then we formulate the problem based on the well-studied private proximity testing.

Scenario

In the scenario, we assume that trajectory-based PCT is used to prevent the spread of COVID-19. We consider a centralized architecture that stores the trajectory data of patients with COVID-19 on a central server and accepts PCT requests from users with their trajectory data. In practice, these patients' trajectories can be received in bulk from public institutions such as a government or health agency.

In the system operation, based on the incubation period of the virus, the server always keeps the trajectory data of the infected patients for the past 14 days [305]. All the data are periodically updated in batches (e.g., once per day at night), with data being added and deleted. The server transforms the trajectory data into an appropriate structure in advance and is always ready to accept PCT requests from clients. A client sends encrypted trajectory data for the past 14 days as a PCT

request, and the server performs contact detection and then returns the results to the client. The results can be time-stamped and signed in SGX as needed so that they can be verified by a third party using an authenticated public key of SGX, allowing clients to use the results for various agencies and events to show that the risk of infection is low. While client data are protected from any other parties except the TEE by encryption, server-side infected people trajectories can be either confidential or open access (i.e., the South Korea case [306]. Our system can be applied to both situations; if the infection data should be confidential, a health agency can encrypt the data before uploading the TEE by RA. Note that the infection data are generally large in size and need to be kept in memory outside of the enclave. This means that the encrypted data initially uploaded to the TEE are encrypted in the enclave and placed in memory outside of the enclave. In our experimental implementation, we consider the confidential case which causes additional decryption overheads.

Problem Statement

Trajectory-based PCT. The trajectory-based PCT protocol is an asymmetric protocol between a client and a server. When a client wants to know the contact with trajectories stored on a server, this protocol returns 1 or 0 to the client depending on the result, and does not disclose the private information of the client to the server. In the use case for infections, each client has a set of trajectory data for one person, and the server has trajectory data for many infected patients.

In conventional private proximity testing [289], when two people, user u and v, have geographic data X_i that consist of location $l_t^{(i)}$ (= (latitude, longitude), e.g., (40.74836, -73.98562)) of time t ($i = u, v$), user u executes the protocol and obtains the following result:

$$\begin{cases} 1 \ \ (\, ||l_t^{(u)} - l_t^{(v)}|| \leq \Theta \,) \\ 0 \ \ (\, others \,) \end{cases}$$

where Θ is a proximity threshold. After that, v does not learn any information about X_u and u does not learn information except whether $||l_t^{(u)} - l_t^{(v)}|| \leq \Theta$.

In a similar vein, trajectory-based PCT can be represented as an extension of this formulation. For contact tracings, a contact can be determined according to human time-series tracking data. We can perform private proximity testing by

extending single geographic data to time-series trajectory data. The threshold can also be extended to 2D thresholds to check the spatiotemporal proximity. PCT allows capturing indirect contacts by examining whether patients are in the proximate place within a specific time period. Therefore, we obtain the following formula denoting the trajectory data of user i as $X_i = (x_1^{(i)} = (t_1^{(i)}, l_1^{(i)}), ..., x_n^{(i)} = (t_n^{(i)}, l_n^{(i)}))$ (e.g., x_{100} =(2020/12/20 12:00:00, 40.74836, -73.98562)), with which the result of contact between u and v can be obtained:

$$\begin{cases} 1 \ \ (\exists\, x_i^{(u)} \in X_u\,,\ x_j^{(v)} \in X_v \, s.t. \\ \qquad ||l_i^{(u)} - l_j^{(v)}|| \leq \Theta_{geo} \text{ and } ||t_i^{(u)} - t_j^{(v)}|| \leq \Theta_{time}\,) \\ 0 \ \ (\,others\,) \end{cases} \tag{7.26}$$

where Θ_{geo} and Θ_{time} are the spatial and temporal proximity thresholds, respectively. Furthermore, v does not learn any information about X_u, and u can obtain only 1 or 0 regarding X_v in this protocol. The thresholds are generally given by medical and epidemiological experts and may be updated. We define this procedure as *trajectory-based PCT*, which can capture indirect contacts by having a certain width in the time direction. Moreover, we can extend the definition so that the duration of exposure to an infected user can be considered, which is recognized as an important factor of COVID-19 transmission [307, 305]. Let Θ_{doe} be the risky duration of exposure, the following formula is obtained:

$$\begin{cases} 1 \ \ (\ \exists\tau \quad s.t. \quad \forall i = (\tau, \tau+1, ..., \tau+\Theta_{doe}) \\ \qquad \exists\, x_i^{(u)} \in X_u\,,\ x_j^{(v)} \in X_v, \quad s.t. \quad ||l_i^{(u)} - l_j^{(v)}|| \leq \Theta_{geo} \text{ and } ||t_i^{(u)} - t_j^{(v)}|| \leq \Theta_{time}\,) \\ 0 \ \ (\,others\,) \end{cases} \tag{7.27}$$

Finding the exact solution of Eq.(7.26) and Eq.(7.27) for the sampled discrete trajectory data is computationally inefficient. Therefore, we simplify this problem by mapping continuous space to discrete space for computational efficiency, where we approximate the PCT problem to the PSI problem. We denote $\mathbb{A}$ as the set of all symbols in a discrete space and $A_i \in \mathbb{A}$ as the i-th element. By mapping $f_\Theta : x \to A$, we can map any point x in the trajectory data to a single symbol A. We call this mapping an "encoding" and introduce the corresponding method in Section B.5. The encoding, f_Θ, must be adjustable according to the granularity parameters Θ_{geo} and Θ_{time}, which correspond to the size of the predefined subspace in the 3D spatiotemporal space, and each subspace corresponds to

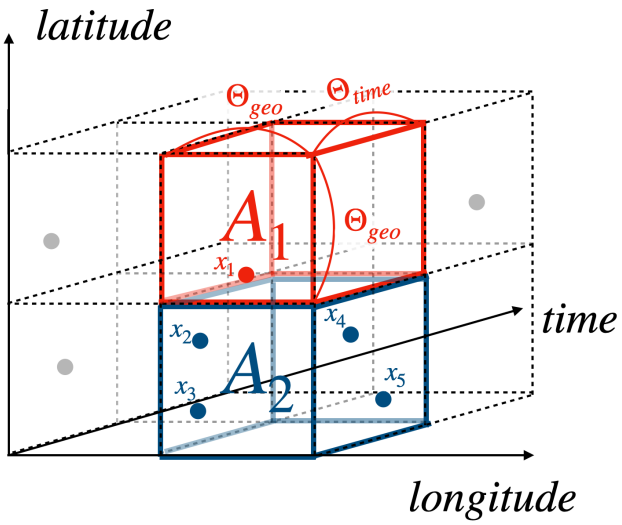


Figure B.2: Spatiotemporal Private Set Intersection.

one unique symbol, as shown in Figure B.2. For example, suppose $f_\Theta(x_1) = A_1$, $f_\Theta(x_2) = A_2$, $f_\Theta(x_3) = A_2$, $f_\Theta(x_4) = A_2$, $f_\Theta(x_5) = A_2$, trajectory point x_1 is mapped to A_1, and x_2,x_3,x_4, and x_5 are mapped to A_2 in Figure B.2. We determine contact by considering the intersection of these symbol sets between u and v. This can be formulated as follows:

$$\begin{cases} 1 \ \big(\exists\, \mathbf{A}^{(u)} \cap \mathbf{A}^{(v)} \neq \emptyset \ \ s.t. \\ \qquad \mathbf{A}^{(u)} = \{\, f_\Theta(x_i^{(u)}) \mid \ x_i^{(u)} \in X_u \,\} \text{ and } \mathbf{A}^{(v)} = \{\, f_\Theta(x_j^{(v)}) \mid \ x_j^{(v)} \in X_v \,\}\big) \\ 0 \ \ (\mathit{others}) \end{cases} \tag{7.28}$$

v does not learn any information about X_u, and u can obtain only 1 or 0 regarding X_v in this problem. We name this problem the *spatiotemporal PSI* or *stPSI* for short. Considering the duration of exposure and given threshold parameters $\Theta = (\Theta_{geo}, \Theta_{time})$ and Θ_{doe}, the problem can be formulated as follows:

$$\begin{cases} 1 \ \big(\, \exists\tau \quad s.t. \quad \forall i = (\tau, \tau+1, ..., \tau+\Theta_{doe}) \\ \qquad (\exists\, \mathbf{A}^{(u)} \cap \mathbf{A}^{(v)} \neq \emptyset \ \ s.t. \\ \qquad \mathbf{A}^{(u)} = \{\, f_\Theta(x_i^{(u)}) \mid \ x_i^{(u)} \in X_u \,\} \text{ and } \mathbf{A}^{(v)} = \{\, f_\Theta(x_j^{(v)}) \mid \ x_j^{(v)} \in X_v \,\})\big) \\ 0 \ \ (\mathit{others}) \end{cases} \tag{7.29}$$

Note that the worst computational complexity this problem causes is the same as in Eq.(3) since we need to sequentially check the u's of all the data only once, both of which are checked $\mathcal{O}(|X_u|)$ times.

In this work, we consider the stPSI as a contact between u and v. Hence, we can reduce the problem to one that is simple enough to be computed with existing confidential computing stacks in a practical computational time. However, the problem is basically an approximation of trajectory-based PCT. Given the threshold parameter Θ, this approximation causes some detection errors. We carefully analyze these errors in Section B.5 and show that there is a trade-off between false positives and false negatives due to the proposed algorithms and thresholds.

Additionally, we describe the important requirements of trajectory-based PCT.
Efficiency. Trajectory-based PCT requires efficiency in several aspects.

- The first is the response throughput since the server will always be exposed to requests from a large number of users. It can be a substantial workload in this centralized protocol.
- The second aspect is the bandwidth. Since the protocol is applied to many users, it is necessary to reduce the bandwidth for communication efficiency.
- The third aspect is scalability. For instance, for COVID-19, the size of the infected patient data and the user data may increase in the event of infection spreading.

The efficiency requirements depend entirely on the context in which PCT is deployed and are determined by the amount of users, frequency of use, number of data, etc.
Security. Security concerns in general PCT systems are manifold. The concerns must include violating the privacy of participants by revealing personal information such as location, inducing errors in PCT results by introducing false information, and violating the integrity and availability of the system. In our scenario, we believe it is reasonable to assume a malicious client, a malicious server, and a semi-honest health agency as the attacker models. Because any client can participate in a service, and because it is not obvious how the server runs in a remote environment, we should assume that it has a full control over the operating system and/or hypervisor, memory hardware units, and packet flow in the network and uses them to attack the system. In addition, the selected health agency is publicly authenticated. In the centralized architecture used in our scenario, some of the attacks that are often of concern in BLE-based PCT methods are not

possible (e.g., carryover attack [275], paparazzi attack [308], Orwell attack [309], etc.). On the other hand, we should consider the following attacks;

- Denial-of-service (DoS) attacks: malicious clients send many requests to the server to bring down the system.
- Query-abusing attacks: privacy violation attacks are performed to obtain an infected person's data by a malicious client's queries.
- Side-channel attacks: the malicious server causes information leakage by side-channel attacks on communication paths and within the server.
- False answering: the malicious server answers clients by sending fake results.
- Fake data injection: the malicious server injects fake infected data into the system.
- Replay: someone catches the client request information by communication interception and reuses it to obtain the client's PCT result.

Our proposed trajectory-based PCT system should have countermeasures to prevent or mitigate all of these attacks.

Flexibility. The rules of risky contact in the case of COVID-19 have been updated as the understanding of the virus has improved [281] as we explained Section B.1. In the trajectory-based PCT, flexibility requires that $(\Theta_{geo}, \Theta_{time}, \Theta_{doe})$ be parameterized and changeable in the PCT system. For example, these parameters need to be changed to minimal values if it is found after the system is released that only direct contact needs to be captured because of the virus's capacity for transmission. However, we do not believe that these parameters need to be parameterized at the client query level (i.e., clients can choose the parameters). In other words, we assume that there is one global condition that is epidemiologically determined to be important for infection prevention. In addition, if the client has control over the parameters, it can expose information about the infected person's data, creating an unnecessary security risk.

We believe that these parameters should be updated only once a day at most. Therefore, it is reasonable to apply the changes in these parameters to the data at the same time as the batch update of the infected data on the server, which justifies our proposed method. In the proposed method, we encode the data in

advance with a granularity that matches the parameters and then use the PSI to detect contact decisions.

Accuracy. Accuracy requires to achieve the high correct detection rate. Since virus infection information is very sensitive, it is necessary to reduce the percentage of responses to clients that contain false positives or false negatives. In addition, when a response is *false*, what kind of *false* it is, i.e., whether it is nearly correct or randomly generated, has completely different meanings in terms of whether there is an upper limit to the error or not. For example, even if the epidemiological false positive is somewhat large, it may be effective in preventing infection depending on the phase of infection [283].

There are several aspects of trajectory-based PCT that can cause a *false* outcome. The first one is the error of trajectory data collection. Many studies have shown that the accuracy of GPS-based trajectory data collection is improved [310, 311]; however, the accuracy is degraded especially in indoor environments. Also, if they stay somewhere for too short a time, they may not be captured when the trajectory data is collected. On the other hand, it is possible to improve the collection accuracy by combining wireless devices such as indoor Wi-Fi [312]. Our work assumes that these state-of-the-art devices collect highly accurate trajectory data, and this false is beyond the scope. The second is a *false* result caused by the stPSI approximation. It is necessary to clarify how many false positives and false negatives are generated by the approximation and what is the upper limit of the error. Through extensive experiments, we empirically clarify how much error is caused by the stPSI. The third is the *false* result caused by the physical environment, which cannot be captured by location information alone. For example, if a person is riding in a car or spending time in an adjacent room, a false positive may occur if only trajectory data are used. This error is difficult to determine from the trajectory data and requires a different type of data; we consider it to be the topic of future work.

B.4 System Overview

We introduce an overview of the system. Table B.3 shows the symbols and parameters that are used in the rest of this section.

Figure B.3 shows an overview of the architecture with a trusted enclave. Our method consists of several steps, including the transformation of data maintained

Symbols	Explanation
N_D	number of raw trajectory data of infected people
D	raw trajectory data of infected people
N_C	number of clients
$c_i \in \mathbf{C}$	a client $i(\in \{1, ..., N_C\})$ and all clients set
N_R	number of chunks of central data
R	mapped D, array of efficient chunks $(= (r_1, ..., r_{N_R}))$
r_i	i-th chunked data of R, efficient representation (e.g., FSA)
q_i	client i's query data (raw trajectory data)
Q	merged and mapped N_C query data (e.g., unique array)
N_Q	unique size of Q
θ	parameter of PCT, $\theta = (\theta_{time}, \theta_{geo})$
(t_i, l_i)	i-th row of trajectory data, time t_i and location $l_i = (l_{ilat}, l_{ilon})$

Table B.3: Symbols and parameters.

on the server side and the transformation of data sent from the clients, as follows.

First, we describe the data of infected patients on the server.

① : (Update master data) The health agency updates the infected patient data D in batch processing. D is in the raw form of trajectory data $D = [(t_1, l_1), ..., (t_{n_R}, l_{n_R})]$ and does not have to include the user IDs since there is no need to distinguish trajectory data by infected users.

② : (Mapping) Step 2 is executed in the same batch processing as step 1. We map from the raw data format D to efficient dictionary representation R with the function mapToChunkedDictionary. This mapping function includes encoding, chunking, and transforming into the dictionary representation. Encoding is to encode each piece of trajectory data into 1D bytes of representation. It corresponds to f_Θ in Eq. (7.28) (7.29). Chunking is to split the dataset into N_R chunks. Transforming is to transform each chunk into a dictionary representation and into R that consists of N_R chunks r_i $(i = 1, ..., N_R)$, where each chunk fits in the enclave memory limitation. How to represent the chunked data specialized in the PSI under the SGX memory constraint is the challenge of this work. These encoding and compression schemes are described in Section B.5. In the case where the infected data are to be kept secret, the encrypted infected data

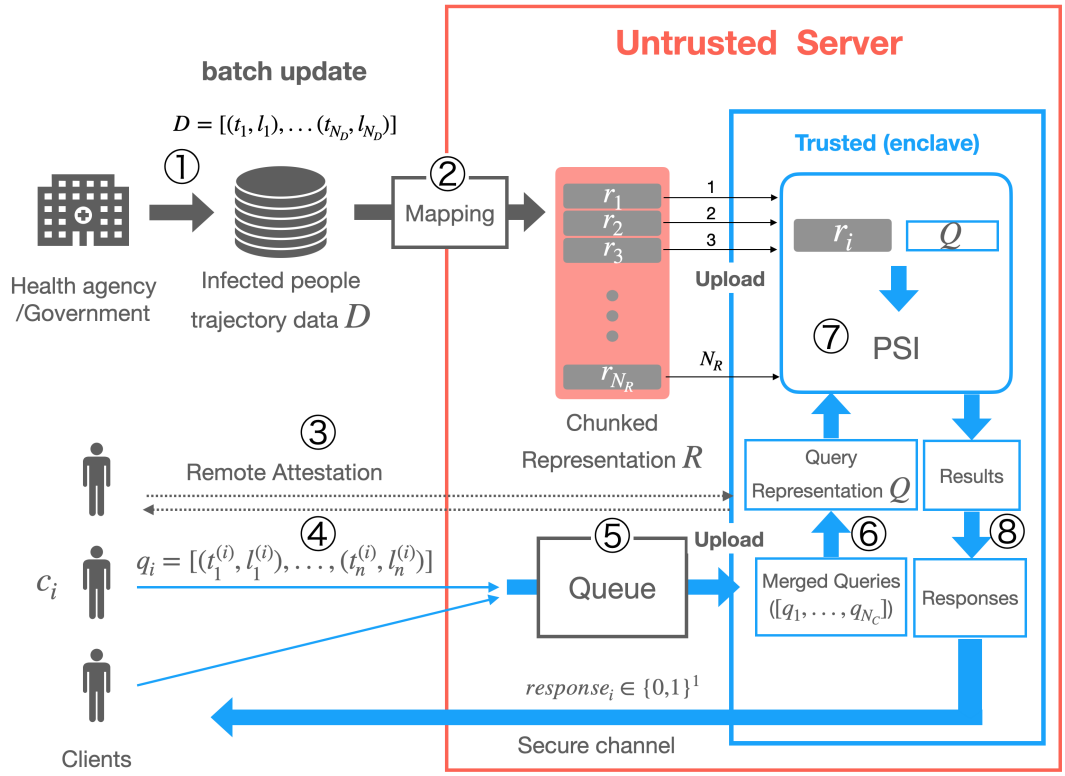


Figure B.3: Architecture overview: circled numbers correspond to the steps shown in System Overview.

are uploaded to SGX by the health agency, step 2 is performed in SGX, and the binary data of the encrypted R are placed in the memory of the untrusted server. The key used for encryption during the upload is obtained from SGX by the health agency through RA.

The next part is the processing of queries from clients.

③ : (Remote attestation) The client verifies the remote enclave through the RA protocol before sending the request to the server. The client can confirm that the enclave has not been tampered with and then securely exchange the key with the enclave. After that, the shared key are used to encrypt the data, which enables secret communication to the remote enclave through a secure channel. Note that a secure channel is a communication channel where only the query data in the request are encrypted and kept secret. Client metadata (e.g., IP address) exchanged in the application layer is not kept secret from the server side and need to be used in the response. The response data are encrypted and kept

secret, as well as the request query data.
④ : (Request) Many clients send PCT requests to the server. In the figure, c_i sends q_i as a request parameter that contains 14 days of her trajectory data. Trajectory data are encoded by f_Θ before encryption; thus, the server and client share the parameter Θ in advance. q_i is encrypted in all the untrusted areas after leaving the client environment and is visible only in the verified enclave.
⑤ : (Queuing) Until a certain number (N_c) of requests are accumulated, q_i is queued outside the enclave, and they are passed to the enclave together by the loadToEnclave function. This function is actually implemented by the so-called *ECALL* to invoke an SGX function. We aim to optimize the loading process for multiple (e.g. 1000) users by batch processing, mitigating the overheads invoked.
⑥ : (Mapping) After uploaded to the trusted enclave, the data are finally decrypted. Inside the enclave, all q_i are grouped together and mapped to query representation Q using mapToArray. While these query data are private and cannot be handled outside the enclave, the size of the enclave memory is strictly limited. Therefore, encoding trajectory data to small bytes is critical.
⑦ : (Contact detection) The chunked data r_i are imported into the enclave one by one, and we compute the set intersection of r_i and Q in the enclave. This can be done by checking the bytes-based match. In the result, only queries with a true result for a set intersection are recorded as positive. If a query is found to be positive, we can reduce some overhead by not computing the PSI for subsequent r_i for the query.
⑧ : (Response construction) After the iterations for all the chunks are completed, responses for all clients are constructed from the results and complete query data q_i $(i = 1, ..., N_C)$ inside the trusted enclave by constructResponses. This process can be carried out by simply encrypting the results (positive or negative) for each client inside the enclave. Finally, the process returns the encrypted result through the secure channel to each client.

B.5 Spatiotemporal PSI

Trajectory Data Representation

In this subsection we focus on the trajectory data representation, which is optimized for PSI processing in the memory constraint of Intel SGX. The most

important issue is how to represent each trajectory data point. This is called encoding and corresponds to f_Θ in Eq. (7.28) (7.29). We need to encode different trajectory data into unique 1D data to solve the trajectory-based PCT as the stPSI, as described in Section B.3. In addition, we need to make the encoded data as small as possible. The compact representation contributes to the whole system performance, and the compressibility contributes to the performance of the PSI part which is the core component of the system.

Additionally, we carefully develop the dictionary representation $R\,(=(r_1, ...r_{N_R}))$ obtained by the mapping in step 2. R should satisfy the following constraints: First, a memory-efficient data structure storing trajectory data should be used to overcome the severe memory constraints of SGX. Second, a fast search should be implemented for a fast PSI. Third, a deterministic search method for an accurate PSI should be provided. The standard dictionary representations do not fulfill these requirements. We consider Hashmap as a baseline. Hashmap ideally supports the $\mathcal{O}(1)$ key-based search. While it provides desirable search performance and allows a deterministic search, it fails to satisfy the efficiency requirements because its size increases linearly with the size of the data. A smaller data structure is preferable in this setting because the overheads caused by SGX are considerably heavy. While probabilistic data structures such as the Bloom filter provide the same search speed performance as that of Hashmap and superior memory efficiency, they do not satisfy deterministic search requirement. It causes random and unpredicted false positives.

Our proposed method to achieve the desired dictionary representation is a combination of effective trajectory data encoding and storing into a fast succinct trie (FST) [290]. An FST is a data structure proposed in [290] and is the base of Succinct Range Filter (SuRF). The SuRF can improve efficiency for queries such as match and range in exchange for false positives, while the FST does not allow any false positives. The FST has basically the same properties as those of trie, but its internal representation is closer to being succinct, and it has a particular strength in spatial efficiency. For more information on the FST, please refer to [290]. Similar to the basic trie, the FST provides high compression performance for highly similar data by sharing a common prefix of encoded bytes in a single node. Essentially, the aim is to ensure that the encoding process transforms trajectory data into highly similar byte sequence representations and then utilizes the similarity to create a compressed dictionary representation using an FST.

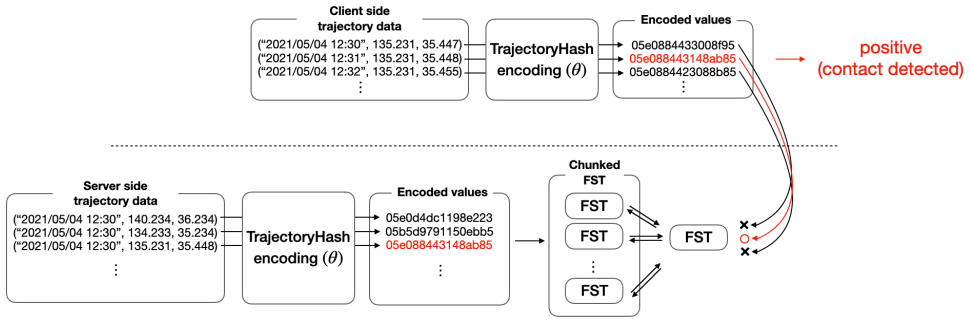


Figure B.4: An overview of running example from raw trajectory to the final risk assessment.

Figure B.4 shows an overview of running example from raw trajectory to the final risk assessment.

TrajectoryHash

We introduce encoding, which corresponds to f_Θ in the stPSI as described in Section B.3. The encoding should satisfy the following 3 properties. First, there is an injective function between different trajectory data and unique byte sequences. Obviously, if this property is not satisfied, the PSI cannot be performed correctly. Second, the encoded value should be small in size. The space after mapping should be as small as possible because if it is small, all data, including both the server data and the queries from the client, will be small. This is the ideal situation for TEE-based secure computation. [313] lacks this aspect; string-level merging caused the binary size of the encoded values to grow. The other desired property is that the encoded values should have many similarities because we can efficiently store within the FST. We introduce *TrajectoryHash* encoding and show that it satisfies all properties.

The trajectory data X consist of an array of tuples of temporal data and geographical data, such as the UNIX epoch and tuples of latitude and longitude, as follows.

$$\mathbf{X} = [x_1 = (t_1, l_1), ..., x_n = (t_n, l_n)]$$
$$t_k \in \text{time (UNIX epoch)}$$
$$l_k = (l_{k,lat}, l_{k,lon}) \in \text{coordinate } ((\text{latitude}, \text{longitude}))$$

Algorithm 17 TrajectoryHash

Require: t,, (l_{lat}, l_{lng}), θ_{time}, θ_{geo}, t_{start}, t_{end}

Ensure: *hash*

$b_1, b_2 \leftarrow$ QuadKeyEncode$(l_{lat}, l_{lng}, \theta_{geo})$ ▷ b_1, b_2 are binary sequence with θ_{geo} length

$b_3 \leftarrow$ PeriodicEncode(t, θ_{time}) ▷ b_3 are binary sequence

$binary \leftarrow$ BitMix(b_1, b_2, b_3)

$hash \leftarrow$ ByteEncode$(binary)$

return *hash*

procedure QuadKeyEncode$(l_{lat}, l_{lng}, \theta_{geo})$

$maxLatitude \leftarrow 360 \arctan(\exp(\pi))/\pi - 90$ ▷ $\simeq 85.05112877980659$

$l_{lat} \leftarrow \min(maxLatitude, \max(-maxLatitude, l_{lat}))$ ▷ for clipping

$px \leftarrow \dfrac{l_{lng} + 180}{360}$ ▷ transform to the Tile Coordinates

$py \leftarrow \left(\dfrac{1}{2} - \dfrac{1}{\pi} \log \dfrac{1 + \sin(l_{lat} \times \pi/180)}{1 - \sin(l_{lat} \times \pi/180)}\right)$

$mapSize \leftarrow 2^{\theta_{geo}}$ ▷ map consists of $2^{\theta_{geo}} \times 2^{\theta_{geo}}$ areas

$x \leftarrow$ Floor$(px \times mapSize)$ ▷ round down function

$y \leftarrow$ Floor$(py \times mapSize)$

$Xbinary \leftarrow$ AsBinary(x) ▷ get as bit array representation

$Xbinary \leftarrow$ ZeroPadding$(Xbinary, \theta_{geo})$ ▷ padding 0 to θ_{geo} length

$Ybinary \leftarrow$ AsBinary(y)

$Ybinary \leftarrow$ ZeroPadding$(Xbinary, \theta_{geo})$

return $Xbinary, Ybinary$

end procedure

procedure PeriodicEncode$(t, t_{start}, t_{end}, \theta_{time})$

$maxLength \leftarrow$ Length(AsBinary$(t_{end} - t_{start})$)

▷ maximum bit length to represent the period t_{start} to t_{end}

$t_{diff} \leftarrow t - t_{start}$

$shift \leftarrow 32 - \theta_{time}$ ▷ 32 = max bit length of UNIX epoch

$t_{diff} \leftarrow$ Floor$(t_{diff}/2^{shift})$ ▷ right shift $t_{diff} >> shift$

$binary \leftarrow$ AsBinary(t_{diff})

$binary \leftarrow$ ZeroPadding$(binary, maxLength - shift)$

return *binary*

end procedure

parameter θ_{geo}	geo distance Θ_{geo}
26	0.6 m
25	1.2 m
24	2.4 m
23	4.8 m

Table B.4: Approximate scale of the parameter θ_{geo}.

parameter θ_{time}	time distance Θ_{time}
32	1 s
26	1 min
24	4 min
22	17 min

Table B.5: Approximate scale of the parameter θ_{time}.

t_1 and t_n are determined as t_{start} and t_{end} considering conditions such as the lifespan of the virus. (e.g., in the case of COVID-19, we currently believe it is 14 days.) Algorithm 17 shows the pseudocode of TrajectoryHash. The input parameters θ_{geo} and θ_{time} correspond to Θ_{geo} and Θ_{time}, respectively. While Θ_{geo} and Θ_{time} directly express the spatial distance as in Figure B.2, note that θ_{geo} and θ_{time} express granularity on a different scale, as shown in Table B.4, B.5. The algorithm of the encoding is based on two encodings QuadKeyEncode, PeriodicEncode and a binary-level mixing function, BitMix. ST-Hash [314] is similar to the encoding. The part to be mixed at the binary level is the same, but the 2 encoding methods and the motivation are different. We use QuadKeyEncode and PeriodicEncode to preserve the trajectory data similarity and hierarchical structure to compress the trajectories.

QuadKeyEncode is based on the quadkey introduced by Bing Map [315], which is a method of encoding into bits in the *tile coordinate* space, recursively dividing into two parts according to a given level, as shown in Figure B.5. Note that in the method, QuadKeyEncode outputs separated binaries. As we can see in the figure, while we obtain "212"(=100110 in binary) using quadkey encoding, QuadKeyEncode outputs 101 and 010. This algorithm is described in detail in Algorithm 17. The parameter θ_{geo} and the approximated distance included in the square in tile coordinates are shown in Table B.4. Strictly speaking, it is correct for both the latitude and longitude at the equator, but the tile length of the latitude is slightly variable according to the height of the latitude, e.g., in New York, $\theta_{time} = 22$ corresponds to $\Theta_{time} = 1.83$. Using this encoding, we obtain unique binaries for each distinguishable area by θ_{geo}. Moreover, we can keep the hierarchical structure and similarity of trajectory locations in the binary rep-

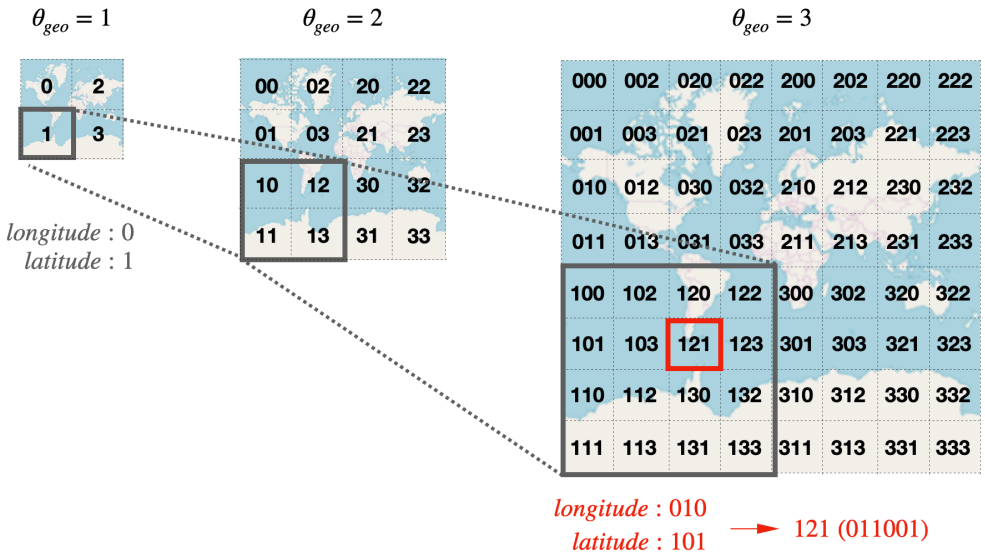


Figure B.5: quadkey and QuadKeyEncode.

resentation. For instance, given $\theta_{geo} = 16$, $(l_{lng}, l_{lat}) = (135.3214557, 30.4564223)$, we obtain the output 1110000000111010 and 0110100100111110 as binaries.

PeriodicEncode is optimized to discretize the time data over a specific given period and at specific given time intervals. This encoding outputs bits with minimum length that can express a distinct time interval according to given θ_{time} in the period t_{start} to t_{end}. Given 14 days as the period, the relation between parameter θ_{time} and the approximate time interval is as shown in Table B.5. The final output length is determined by both θ_{time} and (t_{start}, t_{end}). For example, given $(t, t_{start}, t_{end}, \theta_{time}) =$ ("2020/10/10 10: 00", "2020/10/05 00: 00", "2020/10/19 00: 00", 24) , the processing is carried out in detail as follows:

$t_{end} - t_{start} = 1603065600 - 1601856000 = 1209600$

$maxLength = 21 \ \ (1209600 < 2^{21} = 2097152)$ (Line 21)

$t_{diff} = 1602324000 - 1601856000 = 468000$ (Line 23)

$shift = 32 - 24 = 8$ (Line 24)

$t_{diff} = 468000/2 * *8 = 1828$ (Line 25)

$binary = \text{AsBinary}(1828) = 11100100100$ (Line 26)

$binary = \text{ZeroPadding}(11100100100, 21 - 8) = 0011100100100$ (Line 27)

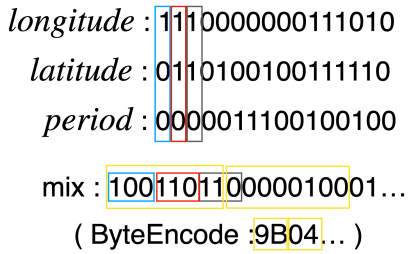


Figure B.6: Mixing TrajectoryHash.

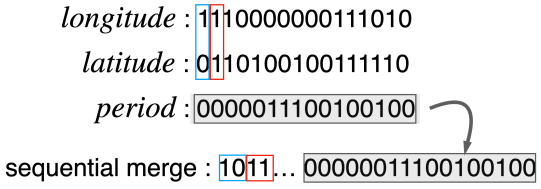


Figure B.7: Sequential merge TrajectoryHash.

Finally, we obtain 0011100100100 as a binary. In this way, we obtain the minimum representation to express trajectory time information and preserve the time representation similarity of the trajectories in the period while adjusting intervals to the given granularity parameter θ_{time}. Obviously, this gives an information-theoretic lower bound on the byte size, since it is the smallest bit representation that identifies a given period. Furthermore, in terms of location information, since the bit array obtained from QuadKeyEncode is also the size of the information-theoretic lower bound for identifying individual regions in the map at the given scale, TrajectoryHash is a succinct representation of the trajectory data for a given granularity.

Now, we have three binaries, i.e., this longitude: 1110000000111010, latitude: 0110100100111110 and periodic: 0011100100100. We mix them into one binary by BitMix (Line 3). We consider that there can be some variants, mixing or simply merging without mixing. A plausible option is to mix one by one from each binary as shown in Figure B.6. In this mixing, the 3D trajectory data are encoded as in Figure B.8, where the 3D similarity of the trajectory data is naturally preserved in the binaries in a balanced manner regarding time and location. By changing the mixing, we can further generalize the encoding. For example, in the encoding of [313], the spatial and temporal information were merged sequentially (Figure B.7) to efficiently share and store the prefixes on the nodes of a finite state automaton.

Last, we encode the bits into bytes by ByteEncode for ease of transport and processing. This may involve extra padding in the prefix to match the size of the bytes. However, since the padding is common in the prefixes, it is largely ignored in the FST.

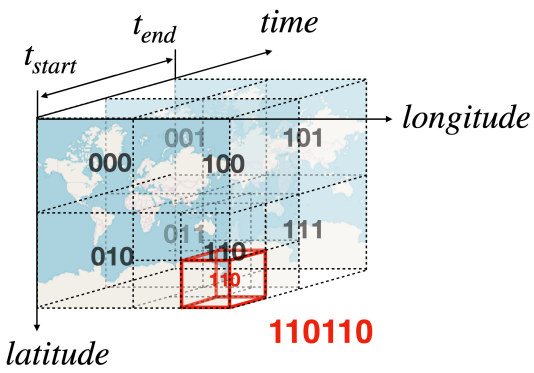


Figure B.8: TrajectoryHash interpretation in the time-series map.

Here, we explain that the FST satisfies the three above-mentioned requirements. We can use the FST as key-based data to store and compress the byte sequence data, sharing prefix bytes from the tree structure's roots. The FST also provides a fast key-based search as a dictionary in proportion to the maximum depth. Moreover, the search cost can be $\mathcal{O}(1)$ if the maximum length is small, which is asymptotically equivalent to the Hashmap and may be advantageous because it does not require computing hash functions. Thus, it basically meets the requirements. In addition, the FST can also provide operations using select and rank to speed up the internal movement between nodes and can efficiently perform multiple adjacent key searches. Therefore, we increase the compression efficiency and search performance for the PSI by introducing the FST, and this data structure satisfies the requirements.

Chunking

We have to consider how to make a chunked FST. At step 2 of Figure B.3, we transform raw data into chunked dictionary representations. Generally, chunking the FST is not a straightforward task because the compression results depend on how we divides the dataset for each chunk, which is different from the Hashmap. With the FST, the greater the similarity of the data contained in each chunk, the more it can be compressed. With the Hashmap, the performance does not depend on these processes because the data size is determined just by the number of stored data. However, we can solve this problem by simple operation. Before constructing the FST, we sort the encoded values in byte order and iteratively take the N_D/N_R trajectory data from top to bottom and transform the data

Algorithm 18 Spatiotemporal PSI

Require: $q_i(i = 1, ..., n_c)$, $\Theta = (\Theta_{geo}, \Theta_{time})$, $R \leftarrow$ MAPTOCHUNKEDDICTIONARY(D, Θ) ▷ ①, ②

Ensure: *Responses*

LOADTOENCLAVE$(q_1, ..., q_{N_C})$ ▷ ⑤ ⑥

$q_1, ..., q_{n_c} \leftarrow$ DECRYPT$(q_1, ..., q_{n_c})$ ▷ by AES-GCM etc. using shared key through RA

$Q \leftarrow$ MAPTOARRAY$(q_1, ..., q_{N_C})$ ▷ ⑥, q_i has list of encoded value and client ID

$Results \leftarrow \{\}$

for $r_i \leftarrow R$ **do** ▷ ⑦ R has N_D chunks

  LOADTOENCLAVE(r_i) ▷ iteratively load chunked data r_i

  $r_i \leftarrow$ DECRYPT(r_i)

  for *query* in Q **do** ▷ ⑦ Q is array with N_Q length

    if r_i.CONTAINS$(query.value)$ **then**

      $Results \leftarrow Results \cup query.clientID$

    end if

  end for

end for

$Responses \leftarrow$ CONSTRUCTRESPONSES$(q_1, ..., q_{N_C}, Results)$ ▷ ⑧

$Responses \leftarrow$ ENCRYPT$(Responses)$

$Responses \leftarrow$ LOADFROMENCLAVE$(Responses)$ ▷ return encrypted data to untrusted area

into a single FST. Then, we obtain N_R chunk of FST. We can stably construct well-compressed FST because a chunk of data has more similarity.

Complexity Analysis

Here, we discuss the asymptotic computational costs of the PSI and precautions. We show the stPSI algorithm for trajectory-based PCT in Algorithm 18. Some of the functions are described in B.4. Dictionary r_i must implement the CONTAINS method, which returns a Boolean value indicating whether the dictionary includes the target or not. In the case of the FST, it is asymptotically constant. The

Algorithm 19 Spatiotemporal PSI (duration of exposure)

Require: $q_i(i = 1, ..., n_c)$, $\Theta = (\Theta_{geo}, \Theta_{time})$, Θ_{doe}, $R \leftarrow$ mapToChunkedDictionary(D, Θ)

Ensure: *Responses*

loadToEnclave$(q_1, ..., q_{N_C})$
$q_1, ..., q_{n_c} \leftarrow$ decrypt$(q_1, ..., q_{n_c})$
$Results \leftarrow \{\}$
for $r_i \leftarrow R$ **do**
    loadToEnclave(r_i)
    $r_i \leftarrow$ decrypt(r_i)
    for q_i in $(q_1, ..., q_{N_C})$ **do**
        $durationOfExposure \leftarrow 0$
        for *query* in q_i **do**
            if r_i.contains$(query.value)$ **then**
                $durationOfExposure++$
▷ add the interval of sampling rate of the trajectory data collection. (e.g., 1 minute)
                if $durationOfExposure \geq \Theta_{doe}$ **then**
                    $durationOfExposure \leftarrow 0$
                    $Results \leftarrow Results \cup query.clientID$
                    break
                end if
                $Results \leftarrow Results \cup query.clientID$
            else
                $durationOfExposure \leftarrow 0$
            end if
        end for
    end for
end for
$Responses \leftarrow$ constructResponses$(q_1, ..., q_{N_C}, Results)$ ▷ ⑧
$Responses \leftarrow$ encrypt$(Responses)$
$Responses \leftarrow$ loadFromEnclave$(Responses)$

computational costs of trajectory-based PCT are as follows. Assume that the cost of a single key search for a dictionary is c and that the unique size of Q is N_Q. The calculation cost is

$$c \times N_R \times N_Q = \mathcal{O}(N_R N_Q)$$

Seemingly, N_Q and the number of chunks N_R is constant, and the PSI is completely scalable for an infected trajectory size. However, note that the size of N_R depends on the memory constraints of SGX. When processing thousands of queries together, the exact $q_i (i = 1, ..., N_C)$ information needs to be kept within the enclave to correctly reconstruct the response, which can be several tens of MB in size; eventually, the size available for chunk r_i is not large. This means that there is actually a practical lower bound on N_R. Last, the routine includes DECRYPT and ENCRYPT. These encryptions are implemented by fast and simple methods, such as 128-bit AES-GCM, and the HW module for encryption is used inside SGX so that the execution time is not dominant.

Algorithm 19 shows the algorithm for considering the duration of exposure. The difference is that Θ_{doe} is required as input, and the number of cases where the PSI is positive are counted continuously for each client (i.e., duration of exposure). As on Line 11, if the server data contain client data, then we add the interval of the sampling rate of the trajectory data collection to $durationOfExposure$. If the server data contain continuous client data, $durationOfExposure$ will be increased; otherwise, it will be reset to zero. Finally, the client is considered positive only when the duration of exposure to risky contacts exceeds Θ_{doe}. Note that this algorithm requires only one scan of the query data; thus, the computational complexity is the same as that of Algorithm 18.

Accuracy Analysis

Here, we analyze how accurate a contact decision by the stPSI (i.e., Eq.(7.28)) is compared to the correct contact decision (i.e., Eq.(7.26)). Since Eq.(7.28) is an approximation of Eq.(7.26), false negatives and false positives may occur. In Figures B.9 and B.10, the central blue point represents one trajectory data point x_i included in the query from the client. The blue rectangle represents the area of the TrajectoryHash value to which x_i belongs (i.e., the stPSI results are positive if there is infected person data in the same area). The red circle represents the area that is judged positive by the exact PCT (i.e., Eq. (7.26)). In

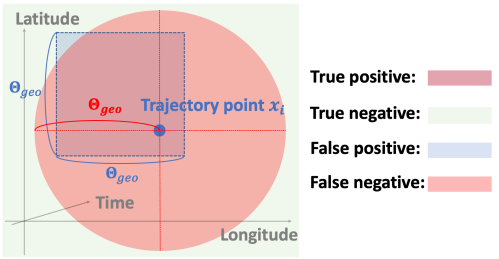


Figure B.9: Possible false outcome in terms of the location.

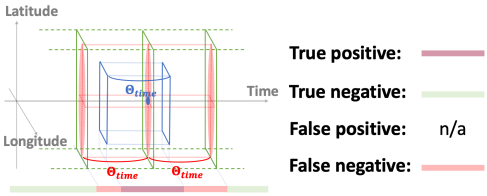


Figure B.10: Possible false outcome in terms of time.

particular, Figure B.9 shows the projection of 3D space onto the longitude and latitude dimensions, indicating a possible false pattern. The false positive occurs in the blue area on the upper left of the blue rectangle in the figure, depending on the position of the trajectory point in the blue rectangle. If there is a false positive, the distance between the false positive data and the trajectory point is between Θ_{geo} and $\sqrt{2}\Theta_{geo}$. A false negative can occur over a relatively large area, as indicated by the red circle in the figure. The possible distance from the trajectory point ranges from 0 to Θ_{geo}. Figure B.10 shows the occurrence of false results in the time direction, where the blue point is the trajectory point, the blue area is the area judged as positive by the stPSI, and the red area is the area judged as positive by Eq.(7.26). In the time-axis direction, since the amount of Θ_{time} that is judged positive by the stPSI is always contained within $2\Theta_{time}$, it is judged positive by Eq.(7.26), as shown in the figure. Therefore, a false positive does not occur, but a false negative occurs. The bounds of the distance between the possible false negatives and the trajectory point are from 0 to Θ_{time}. Note that these analyses are for individual data points. In an actual contact detection query, we have a large amount of infected person trajectory data. Therefore, these false outcomes may not be very problematic because the query will return 1 if at least one positive sample is found in the entire infected person dataset. In Section B.6, we perform an empirical evaluation of these false outcomes by experimenting with the trajectory data and show that the resulting detection is highly accurate.

False negatives can occur in the stPSI, which can be problematic. False negatives in PCT may encourage potentially infected individuals to become active, which may contribute to the spread of infection. In addition, as shown in the pre-

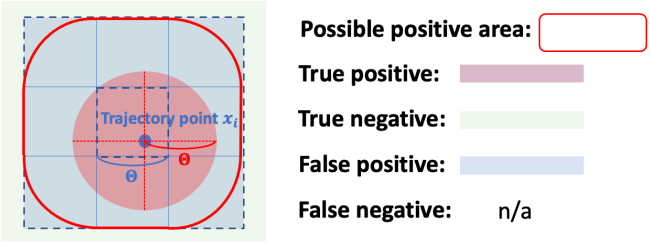


Figure B.11: *nfp-stPSI* design (2D version for simplicity).

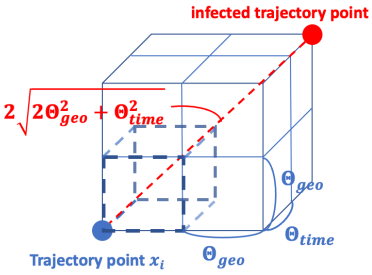


Figure B.12: Maximum distance between trajectory point and false positive.

vious analysis, the minimum distance between the trajectory point and the false negative data point obtained by the stPSI is 0. On the other hand, false positives may be effective in spreading infection. Pandl et al. [283] shows the importance of appropriate selection of the PDR, suggesting that a wider PDR, i.e., the allowance of false positives, may help prevent the spread of infection. Therefore, we believe that a false positive is more acceptable than a false negative in PCT. We devise a variant of the stPSI, *nfp-stPSI* (non-false negative-stPSI), which has only false positives and no false negatives. Moreover, there is a guarantee that the distances between the trajectory point and false positive samples are tightly bounded.

nfp-stPSI is a simple extension of the stPSI that intersects all blocks around a block trajectory point belonging to a 3D encoded space (Figure B.1). In other words, the search is performed on 26 surrounding blocks in addition to the original center block to which the trajectory point belongs. The hash values of these blocks can be efficiently computed from the binary of the trajectory point's TrajectoryHash by some bit operations. Figure B.11 shows the design of the nfp-stPSI in 2D space for simplicity. nfp-stPSI judges all the surrounding blocks as positive. Therefore, the exact positive area according to Eq. (7.26), indicated by the red circle in the figure, is always detected as positive. The area shown by the red bold line in the figure includes all possible exact positive areas (Eq.(7.26)) for the trajectory points belonging to the center block, and it is always included in the area of the external blue blocks. Hence, in nfp-stPSI, false negatives do not occur. As shown in Figure B.12, the maximum distance between the false

positives and the trajectory point is reached when the trajectory point belongs to the inner corner of the center block and the infected data are in the outer corner of the external block, with a maximum value of $2\sqrt{2\Theta_{geo}^2 + \Theta_{time}^2}$. Therefore, the false positive that occurs is guaranteed to be close to the correct answer to some extent. Thus, although nfp-stPSI increases the overhead of the search, it eliminates false negatives and generates acceptable false positives. In the next section, we evaluate this method empirically.

B.6 Experiments

We conduct experiments using real trajectory data to demonstrate that the proposed architecture for PCT can achieve high query throughput and the expected properties.

Experimental setup. We use an HP Z2 SFF G4 Workstation, with a 4-core 3.80 GHz Intel Xeon E-2174G CPU (8 threads, with an 8 MB cache), and 64 GB RAM, which supports the SGX instruction set and has 128 MB processor reserved memory (PRM) in which 96 MB EPC is available for user use. The host OS is Ubuntu 18.04 LTS, with Linux kernel 5.4.0-72. We use version 1.1.3 of the Rust SGX SDK[l] [316] which supports Intel SGX SDK v2.9.1, and Rust nightly-2020-04-07. Our experimental implementation is available on Github[m].

Preliminary Experiments

Before the experiments, as described in Section B.2, we consider the secure-hardware-based PSI to be much better than the cryptography-based PSI in terms of efficiency. To verify this, we compare both PSI executions under setting similar to the scenario. For fairness, we compare single end-to-end PSI query without multiplexing optimization, as described in Section B.4. Our secure-hardware-based approach implementation is based on Intel SGX and simply uses Hashmap and performs the PSI inside the enclave; the OT-based approach implementation [n] follows [287]. Table B.6 describes the execution time comparison between the OT-based [287] and secure-hardware (Intel SGX) -based PSI under the balanced setting, where we assume that only the RA protocol is performed in advance and

[l] `https://github.com/apache/incubator-teaclave-sgx-sdk`
[m] `https://github.com/ylab-public/PCT`
[n] `https://github.com/osu-crypto/libPSI`

size N (bytes)	**Intel SGX** execution (ms) / communication (MB)	**OT** [287] execution (ms) / communication (MB)
10^3 (16 KB)	38 / 0.016	35 / 2
10^4 (0.16 MB)	52 / 0.16	207 / 22
10^5 (1.6 MB)	153 / 1.6	2389 / 235
10^6 (16 MB)	1552 / 16	27110 / 2482
5.0×10^6 (80 MB)	121526 / 80	154826 / 12502

Table B.6: PSI comparison: cryptography-based vs secure-hardware-based; execution time (balanced)

that the online phase includes the client-side encryption and decryption time. We change the set size 10^3 to 10^6, and each data point has 128 bit. As shown in Table B.6, Intel SGX can easily overcome the state-of-the-art method in the balanced setting. In particular, at 10^5 10^6 the difference in execution time becomes significant because of the overhead of oblivious transfer while SGX has scalability in this range of sizes. Additionally, the secure hardware substantially improves the bandwidth. The communication cost of SGX is almost the same as the original size because the data we have to send are just data encrypted by a symmetric key such as AES-128. Assuming many clients, this condition is essential. Although the efficiency is better in the two aspects, we should also pay attention to the last line in the table. Despite using Intel SGX, the execution time is very slow because of the memory constraint of SGX. When $N = 5.0 \times 10^6$ (80 MB), the trusted enclave has to handle approximately 160 MB of data, which exceeds the memory limitation (=96 MB). As a result, serious overhead occurs.

Table B.7 shows the results obtained when using Intel SGX in the unbalanced setting. We also show the results of [64] as a reference [o] i.e., the total execution time (online sending and receiver's encryption and decryption) of the best parameters and the maximum multithreading (≤ 64). These numbers are the best of their implementation, but the table shows Intel-SGX-based PSI is significantly fast and efficient in the unbalanced setting, even though the security model is stricter than [64] (semi-honest). The secure-hardware-based PSI is basically not

[o] [64] does not offer open-source implementation; thus we directly compare with the results they reported.

size N (bytes)	size n	**Intel SGX** execution (ms) / communication (MB)	**HE [64]**
2^{16} (1 MB)	5535	77 / 0.089	600 / 2.6
2^{16} (1 MB)	11041	73 / 0.17	1300 / 4.1
2^{20} (17 MB)	5535	72 / 0.089	2200 / 5.6
2^{20} (17 MB)	11041	85 / 0.17	4000 / 12.0
2^{24} (268 MB)	5535	249 / 0.089	10600 / 11.0
2^{24} (268 MB)	11041	424 / 0.17	16200 / 21.1

Table B.7: PSI comparison: cryptography-based vs secure-hardware-based; execution time (unbalanced)

affected by the server-side data size N, as shown in Table B.2. However, when it is beyond the SGX memory constraint, the execution time becomes slow due to paging overheads, as shown at $N = 2^{24}$ (268 MB). In this case, the client size is very small, and less paging is required, and the impact is smaller than that of the previous result.

In this way, we can achieve a fast PSI by utilizing secure hardware. We expect cryptography-based methods to gradually improve; however, it is unlikely that they will catch up to the secure-hardware-based method in the near future. For deployment in a practical situation for the PCT system, it is better to adopt secure hardware.

Experiments

Datasets. We conduct the experiments with a synthetic dataset and real datasets. The synthetic dataset is generated by the density EPR model [317] implemented in scikit-mobility [p]. We extend this implementation to describe a more continuous human mobility; it is reproducible by our open-source code [q]. The data are individual trajectories for every minute of 14 days in New York City. The real dataset includes data on people's trajectories in specific regions of Japan avail-

[p] https://github.com/scikit-mobility/scikit-mobility

[q] https://github.com/ylab-public/PCT/tree/master/tools/trajectory

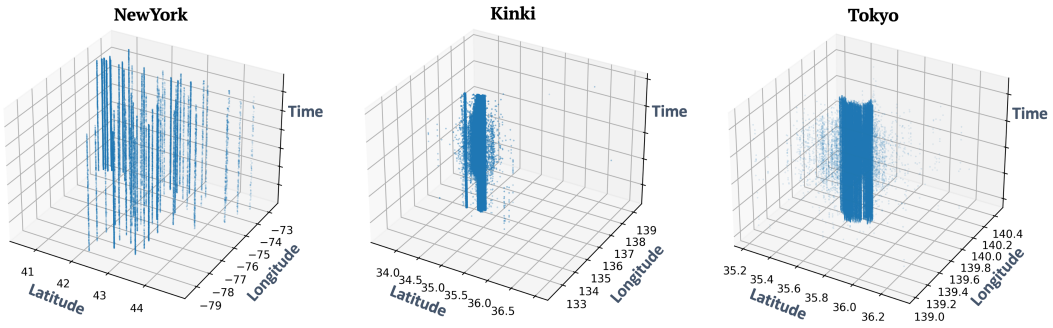


Figure B.13: Trajectory distributions of New York, Kinki and Tokyo. Note that the latitude and longitude scales are different.

able in JoRAS[r] of The University of Tokyo. We use the people flow dataset[s] for Kinki[t] and Tokyo[u] in Japan to create the experimental dataset. Note that this dataset has individual trajectories for every minute of only a single day for privacy. We extract only the time and coordinate information and create the dataset by applying the encoding described in Section B.5. Figure B.13 shows this 3D trajectory data distribution of New York, Kinki and Tokyo from left to right. The scatter is 100000 trajectory points randomly sampled. Since the New York data are generated by EPR model [317], they are distributed in approximately 70 regions. Compared to Kinki, Tokyo has a much denser distribution.

Accuracy Experiments

To empirically evaluate the contact detection accuracy of the stPSI and nfp-stPSI, we prepare the server-side infected person data and client-side query data for the New York, Kinki and Tokyo datasets. Table B.8 shows the dataset scale. To prepare these data, regardless of whether they were clients or servers, for the New York dataset, we randomly generated trajectories for each user as above-mentioned, and for the Kinki and Tokyo datasets, we randomly sampled users without replacement from original data. We measure the results of queries by

[r] http://www.csis.u-tokyo.ac.jp

[s] Precisely, these people flow datasets are synthetic datasets that are elaborated from real trajectories. More details about the specific process can be found here: http://pflow.csis.u-tokyo.ac.jp/data-provision-service/about-people-flow-data/ and/or [318]. In this work, we consider these datasets as real datasets.

[t] https://joras.csis.u-tokyo.ac.jp/dataset/show/id/3038201000

[u] https://joras.csis.u-tokyo.ac.jp/dataset/show/id/3000200800

	client query data	server infection data
NY	20160×100	20160×1000
Kinki	1440×100	1440×14000
Tokyo	1440×100	1440×14000

Table B.8: Data set size used in accuracy evaluation. For example, New York's client query data has 100 clients and each of them has 20160 trajectory point records (e.g., every minute for 2 weeks), totally 2016000 records for client query data.

the stPSI and nfp-stPSI and compare them with the correct answers that exactly satisfy Eq. 7.26 for each client. The result of each query is obtained as a binary.

Tables B.9 and B.10 show the results of the stPSI and nfp-stPSI, respectively. The tables show the parameters used for the stPSI and the true positive, true negative, false positive, and false negative rates of the contact detection. Each value represents a percentage and (the number of queries). The stPSI has a detection accuracy (true positive + true negative) of approximately 90% for the New York dataset and a higher accuracy, closer to 100%, for the real Kinki and Tokyo datasets. However, looking at the New York dataset, the stPSI shows nonnegligible false negatives, as theoretically expected, and the recall is higher, especially as the threshold parameters become more granular. Compared to the stPSI, nfp-stPSI improves the false negative rate and accuracy for all datasets and causes false positives. Especially, in the case of New York and $(\theta_{geo}, \theta_{time}) = (25, 25)$, nfp-stPSI greatly improves on the true positive and false negative rates, with a relatively small false positive rate. Moreover, all of these false positives are "close" errors.

Overall, the accuracy is high, especially for real data. Even when the true positive and true negative rates are extremely high in Kinki and Tokyo, respectively, the detection accuracy is high in both cases, which emphasizes that our proposed method is significantly effective. In the case of real data, we believe that infected people are moving continuously and are more likely to be in contact with multiple points than with a single trajectory data point; hence, the contacts can be captured even with a fixed block area in the stPSI or nfp-stPSI.

We implement the method in [267] and experiment with its accuracy. The method performs PCT based on trajectory data, similar to our method, but con-

	θ_{geo}	θ_{time}	TP	TN	FP	FN
NY	21	21	49% (985635)	37% (745317)	0% (326)	14% (284722)
	24	22	29% (575880)	61% (1224748)	0% (196)	11% (215176)
	25	25	9% (178467)	80% (1611751)	0% (246)	11% (225536)
Kinki	24	22	1% (1007)	99% (141409)	0% (0)	0% (45)
Tokyo	24	22	78% (112459)	20% (28145)	0% (2)	1% (1885)

Table B.9: *st-PSI* can achieve high accuracy for all trajectory dataset, but cause false negatives.

	θ_{geo}	θ_{time}	TP	TN	FP	FN
NY	21	21	63% (1270303)	26% (530334)	11% (215309)	0% (54)
	24	22	39% (791054)	55% (1104383)	6% (120561)	0% (2)
	25	25	20% (404003)	74% (1489927)	6% (122070)	0% (0)
Kinki	24	22	1% (1052)	99% (141380)	0% (29)	0% (0)
Tokyo	24	22	80% (114313)	19% (26772)	1% (1375)	0% (1)

Table B.10: *nfp-stPSI* improves accuracy and remove false negatives, but cause false positive.

siders only the location information and not the temporal proximity. Therefore, we randomly select 100 time points out of 14 days and apply their method on the trajectory data of those time points. In their method, each data point is rounded to the closest position on the grid, where the distance between the points on the grid is set by θ_{geo} and θ_{time} in the experiment. As shown in Table B.11, their method can generate false negatives as well as the stPSI can.

Note that in these accuracy evaluation experiments, the accuracy of collecting trajectory data including GPS tracking is beyond the scope. The detection accuracy we are discussing is focused only on the error generated by the algorithm, not the error caused by the radio signal troubles, and so on.

Performance Experiments

We mention two notes about the performance experiment as a whole. First, note that in the following experiments, both the client and server send trajectory data "every minute" for two weeks. In practical scenarios, the data interval may be

	θ_{geo}	θ_{time}	TP	TN	FP	FN
NY	24	22	22% (2206)	70% (6986)	0% (2)	8% (806)

Table B.11: The method of Reichert et al causes similar errors for sub-sampled NewYork dataset.

larger, and hence the client query size will be smaller. Therefore, the number of clients and infected people are not very important in these experiments. Second, in measuring the execution time, we report the worst case of the stPSI, i.e., while we can avoid checking queries that have already been judged as positive in previous chunks during the execution of the stPSI, since the stPSI depends on the distribution of the data, we do not skip this step and always calculate the intersection for all query data. This is important in terms of security and can be a countermeasure against side-channel attacks, which guess the query result from the execution time. Hence, the execution time practically be shorter than the experimental result as usual.

First, we show the compression results. In our method, the trajectory data are encoded by TrajectoryHash and stored in the FST. The baseline method is to use TrajectoryHash and Hashmap (of the state-of-the-art Rust standard library [v]). Figure B.14 shows the sizes of the dictionary data representation for TrajectoryHash-encoded trajectory points stored in the FST and Hashmap. The dataset used here is the server-side dataset shown in Table B.8, and Random is random data instead of trajectory data. NY (24-22) corresponds to the New York dataset with TrajectoryHash parameters $(\theta_{geo}, \theta_{time}) = (24, 22)$. The results show that, overall, the FST is able to efficiently compress and keep the data encoded by TrajectoryHash compared to the Hashmap. For many datasets, the compression is more than approximately 5 times better. Therefore, we can see that compression can be achieved regardless of the granularity of the parameters or density of the trajectory data. The fact that the compression efficiency for the actual trajectory data is greater than that for the random data confirms that TrajectoryHash preserves the similarity of the trajectory data and that the FST is able to compress the data well. The reason why the Hashmap size is different for the same data size is that some of the trajectory data can be grouped into the same TrajectoryHash value (e.g., continuous observations at the same location

[v] `https://github.com/rust-lang/hashbrown`

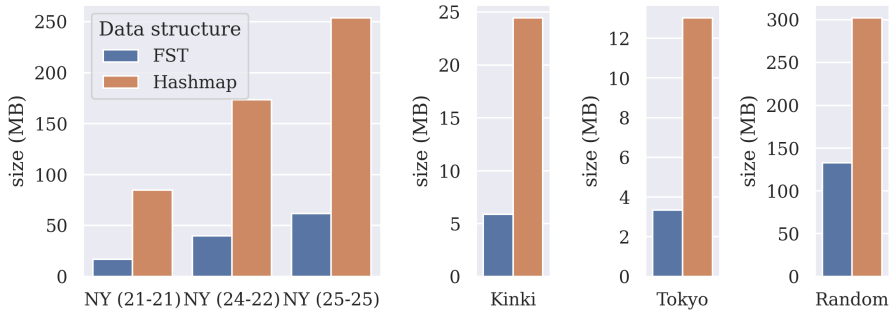


Figure B.14: FST can efficiently keep trajectory data encoded by TrajectoryHash for different datasets.

within a few minutes). This result supports the fact that the combination of TrajectoryHash and the FST can perform PCT on a large amount of data, even under the tight memory constraints of SGX.

To operate the system efficiently under the strict enclave memory constraint, we have to determine the proper chunk size. We experiment to find the proper chunk size, using the New York dataset and encoding with $(\theta_{geo}, \theta_{time}) = (25, 25)$. The server-side data size is 20160×5000, and the client-side data is 20160×200. If the chunk size is 1, i.e., without chunking, the server-side data size is approximately 282 MB, which causes very large overhead in allocating memory in the enclave area because the process tries to allocate virtual memory using special paging beyond the EPC size. We address chunking the FST by presorting the trajectory data, dividing the data into chunks, and converting each chunk into an FST as described Section B.5. Figure B.15 shows the data for each chunk when the above-introduced server data are divided into 10 chunks. The sum of the data size is slightly less than the original data size of 282 MB. We consider each FST to keep data with high similarity by sorting before chunking.

Then, we evaluate the relation between stPSI performance and chunk size. Figure B.16 shows the PSI execution time for different chunk sizes when using the FST and Hashmap. The results of the FST show that the execution time decreases as the chunk size increases, being fastest around 20 to 30, and becomes slower as the chunk size increases. Compared to the result of the Hashmap, the FST can achieve better performance overall. When the chunk size is small, the execution time is affected by the memory constraints of SGX, and when the chunk size is large, the execution time is affected by the cost of the repeated ECALL,

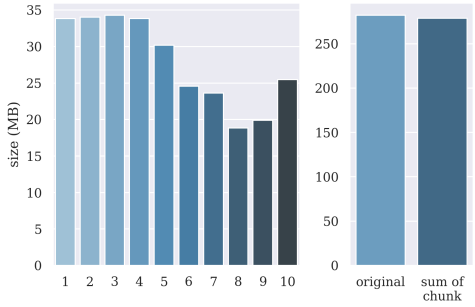


Figure B.15: Sizes of 10 chunks for 282 MB server data, and original data size and sum of these chunks.

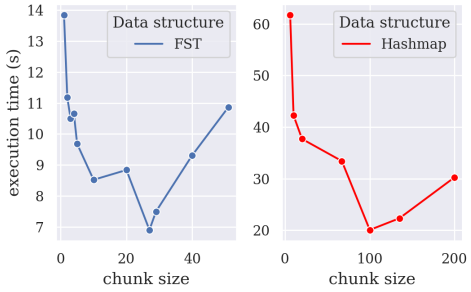


Figure B.16: The execution time varies depending on the number of chunks.

which is function used to upload data into the enclave and switch the process to a trusted process. Considering that the client query data size is approximately 32 MB (= 8 bytes × 20160 × 200), the results are reasonable since the EPC limitation is 96 MB. For example, when the chunk size is less than 5, the execution time increases despite the small number of iterations because each chunk is more than 56 MB (=282 MB /5), which incurs a paging cost. However, when the chunk size is small (e.g., less than 5), there is not such a large overhead even when the EPC memory size is exceeded. This outcome occurs because the internal implementation of the FST is such that the data to be accessed belong to a contiguous memory area and the number of paging occurrences is small. In the case of the Hashmap, the overhead is more outstanding. Although not included in the figure, when the chunk size is 1, the execution time is 532 seconds. The size of one chunk when divided into 10 chunks is roughly 100 MB, and if divided into 50 chunks, it is approximately 20 MB. Compared to the FST, since the memory layout of the data of the Hashmap is scattered, the paging of the EPC is likely to occur frequently.

Next, we evaluate the overall performance. Figure B.17 shows the overall execution time of the PCT when the server data are fixed at 5000×20160 and the client data increase from 100×20160 to 500×20160. The figure on the left shows the execution time of the stPSI, and the figure on the right shows the execution time of nfp-stPSI. The execution times of the four phases of the stPSI, i.e., computation of intersection, decryption of the server data, decryption of

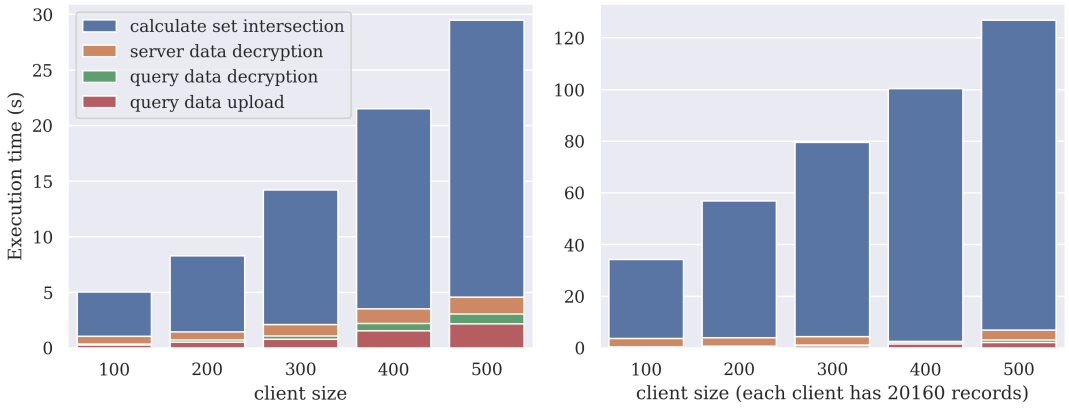


Figure B.17: The stPSI can achieve high performance on a practical data scale (left). nfp-stPSI causes a larger overhead (right).

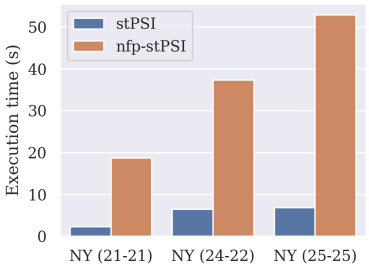


Figure B.18: Finer granularity of parameters directly degrades the performance.

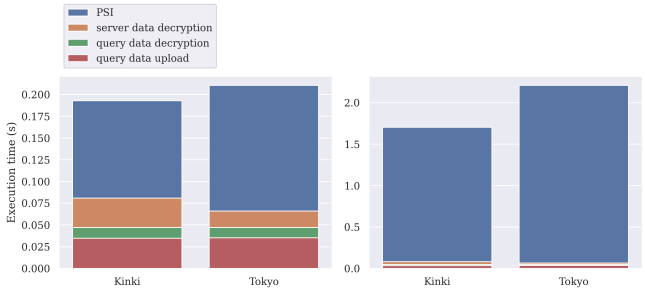


Figure B.19: Results of the stPSI (left) and nfp-stPSI (right) on real data.

the query data, and uploading of the query data to the enclave are shown in different colors. In the stPSI, the number of chunks is set to 20, and in nfp-stPSI, the number of chunks is set to 1. These values are selected because the number of searches to the FST increases enormously in nfp-stPSI; thus it is more effective to reduce the number of chunks to reduce the number of iterations. The execution time of the stPSI and nfp-stPSI increases almost linearly with the client data size. As the client size increases, the time required for uploading and decoding the query data becomes nonnegligible, but the computational cost of the intersection is completely dominant. Compared with the stPSI, nfp-stPSI

causes a large overhead, approximately 4 to 6 times larger. This result indicates that there is room for further improvement in the implementation of nfp-stPSI. Overall, however, our proposed algorithms are able to return results with practical execution times for data with a scale close to that of real data.

We compare the results in terms of performance with [267], which also achieves PCT by secret computation, similar to our method. Their method performs the secret computation by ORAM [319]. We perform PCT by using the same implementation of binary search with Floram[w] as in their work. We assume that the server data consist of 1000 trajectories and the client data consisted of 100 trajectories[x]. In their algorithm, the precomputation increased the size of the data by a factor of 9, resulting in the need to compute the intersection of 9000 points of server data and 900 points of client data. Even though this is a setting that is approximately six orders of magnitude smaller than the scale of the experiment, the computation takes 123.3 seconds on average. This result indicates that the overhead of ORAM is still unacceptable at present and that more emphasis should be placed on the speedup of the secret computation that the TEE brings.

Figure B.18 shows a comparison of the execution time for several granularity parameters. The used client size is 200×20160, and the server size is 5000×20160. Because the parameters are set to finer granularity, the execution time essentially increases. This phenomenon occurs due to the increase in the size of the hash value and the decrease in the compression ratio by the FST due to the decrease in the similarity and coincidence rate of the trajectory data. The size of the hash value for $\theta_{geo}, \theta_{time} = (21, 21)$ is 7 bytes, but for (24,22) and (25,25), it is 8 bytes.

The results of the stPSI and nfp-stPSI execution times for the real Kinki and Tokyo datasets are shown in Figure B.19. The execution times shown are the same as in Figure B.17. The used client data size is 1440×200, and the server data size is 1440×14000. In the stPSI, the percentage of the execution time for decryption and query data upload is relatively high, while in nfp-stPSI, the percentage of intersection computation is high. Overall, the execution time is kept small, which shows that our proposed algorithm works fast even for real data.

[w] `https://gitlab.com/neucrypt/floram`

[x] Note that this is different from the experimental scale of our proposed method because the computation of the competitor is too time consuming.

Requirements Analysis

We show how the system meets the requirements we introduced in Section B.3.
Efficiency. There are four elements that make the system efficient: Intel SGX, chunking, data representation, and query multiplexing. First, SGX bring us an efficient PSI. SGX allows software to perform secret computations transparently and eliminates the need for complicated and time-consuming cryptographic techniques to perform the PSI. This fact can be confirmed the preliminary experiments and is the main basis of the efficiency of the system. The computational overhead is small, and also bandwidth is dramatically improved compared to existing methods. Second, chunking R into r_i $(i = 1, ..., N_D)$ avoids the serious paging overhead caused by the severe memory constraints of SGX even when the data of infected patients become too large to fit into the enclave. Third, the memory-efficient dictionary representation (Section B.5) reduces the number of chunks, resulting in reduced PSI execution and overheads for upload to the enclave. This is also an important element of the system. Fourth, steps 5 and 6 (Section B.4) show query multiplexing and improve the throughput of the query processing. Reading the chunked data r_i, as in Step 7, is costly due to the N_D iteration, and doing this for every query yields large overheads. As a result, the system achieves high query throughput and scalability.
Security. Our protocol follows the RA and secure computation provided by Intel SGX. Previous studies [320, 304] show the protocol security. Informally, any state cannot be observed from outside the TEE, and even if any inputs are known, any tampering with the state that can be performed by the malicious server will not divulge any information about the client trajectories. Hence, it is guaranteed that all information an attacker can observe is only outside the TEE. However, in this system, all information observed outside the TEE must be encrypted. Therefore, cryptographically strong security for the client's privacy from any external attacker is ensured when using proper encryption. More formal definitions require elaborate modeling of the attacker and private information, but our setting is common, and we defer to earlier work [320, 304]. Note that some side-channel attacks, such as Spectre [182] and other cache attacks, are beyond the scope of their work and our work. To protect against these attacks, we have to consider *data-obliviousness* [321, 322] to make the side channels uniform.

Considering the risk analysis discussed in Section B.3, some simple extensions are needed. DoS attacks: To prevent DoS attacks against SGX, it is necessary to

authenticate the client before SGX processes it. Since the data on the server side change only once a day, each client should be allowed to make only one request per day. This can be done with a simple extension, since the information of the user requesting the query does not need to be anonymized. Query-abusing attacks: This can also be done by authenticating the client to minimize the damage of attacks. The client can obtain only the 0 or 1 information. However, even if the request is limited to once a day, there is a possibility that some information may be leaked as a result of using multiple compromised clients. To prevent this, other protection mechanisms are necessary; for example, assuming the possibility of linking the obtained information with other information, preserving DP should be considered. For example, recent work [323] proposes a variant of the DP that is relevant for this purpose. Side-channel attacks: Since the data are encrypted by a key shared between SGX and the client, it cannot be leaked from the network. On the other hand, attacks through the side channels inside the server may be possible due to cache attacks, etc. Defending against these attacks is the topic of future work. False answering: False answering is prevented because the enclave's internal blogs are verified by RA. Fake data injection: In this scenario, we assume that the health agency is a trusted institution, and when a third party injects data, it is necessary to modify the data in the memory encrypted by SGX. Thus, it is cryptographically protected. Replay: A replay attack can occur even when a secure channel is in place, but it can be completely prevented by authenticating the user, restricting the number of requests per day, and updating the key every day.

Flexibility. We achieve flexibility by parameterizing the encoding of the data using a granularity parameter Θ. The three parameters we introduce are geographic granularity, temporal granularity, and duration of exposure to the virus. The parameters are shared between the server and clients in advance. We have to update all the data when we want to change the rules of risky contact. In other words, our method cannot change the flexibility parameters more frequently than batch updates. Essentially, we believe that these parameters are determined based on epidemiological arguments and that their change frequency is not more than the frequency of batch updates of the data (i.e., once a day).

Accuracy. Our proposed method may generate nonnegligible false negatives and false positives. However, we have shown that it is possible to extend the method to generate only false positives with a reasonable upper bound on the error. Note

that we have focused only on the errors caused by the algorithm. In the real world, in addition to this, there are errors that occur during data collection and errors that occur due to the physical characteristics of buildings and vehicles. To perform contact detection based on this information, more metadata and other information may be required. There is a trade-off between the cost of information collection, privacy, and accuracy. A more detailed discussion of the trade-offs involved from epidemiological perspectives is beyond the scope.

B.7 Related Works

There are DP3T [135] and similar schemes [275, 276, 274, 273], which are BLE-based decentralized architectures that use a device's wireless signals. They are the most popular implementation methods to date. The major difference from our proposed system is that while they handle only contact information through random ID tracking, we directly handle trajectory data in a private manner to detect indirect contacts. Desire [324], known as hybrid architecture, has basically the similar characteristics as decentralized architectures. In Desire users send random ID-based contact information to the server, but the ID cannot be used to identify an individual. Therefore, there is no way to discover an indirect contact based on trajectory data.

Hamagen [325] and AAROGYA SETU [y] are similar applications developed for COVID-19 but use trajectory data. Hamagen uses trajectory data to identify the contact location, but the logic of contact determination is different from that of our proposed method because Hamagen is a BLE-based method similar to DP3T. In other words, Hamagen does not attempt to use trajectory data to detect contacts but rather to publish the locations where contacts with infected people have occurred. AAROGYA SETU collects raw trajectory data, unlike the proposed system; thus, there are serious privacy concerns [326]. There are other studies [327, 328] that propose the use of richer data such as users' detailed trajectory data. In contrast to these studies, our method differs in that we use these rich trajectory data to find instances of indirect contact.

We summarize the comparison between the BLE-based PCT technique and our method (centralized architecture) as follows. First, from the security and privacy points of view, the characteristics are very different, as described in Section

[y] `https://www.aarogyasetu.gov.in/`

B.3 and [329]. In centralized architectures, it is important to discuss the trust model. We have to consider how to protect a client's privacy when the server is untrusted. In the BLE-based approach, there is no such security concern since personal data are not exposed to any party. On the other hand, BLE-based methods have security risks that do not arise in a centralized architecture, such as continuous tracking of random IDs. In most cases, to ensure the security of a centralized architecture, either a very costly method such as secret computation or a strong assumption such as trusting the server is required. This is one of the main reasons why a decentralized architecture is preferred. Second, in terms of efficiency, as opposed to the BLE-based method, where the contact decision is computed on each device, our method is computed on the server, which can be computationally intensive. The advantage of the BLE-based method is that the computational cost is lower than that of our method, which compares the data of many infected people, because the target of the computation is limited to only those who have direct contact. On the other hand, it is not possible to detect, e.g., indirect contact, via the BLE-based method. In terms of communication cost, the BLE-based method basically requires broadcasting of the infected person data. While the size of these data is very small, it is necessary to notify all users. In a centralized architecture such as mine, users need to send information about their trajectory data only when they query the server, but the data size is much larger. Finally, regarding storage, while the BLE-based method stores only the contact's ID, our method requires the user to store the trajectory data on his own device. Third, we compare the ability to detect contacts. We mentioned that our method is able to detect a wider class of contacts (i.e., indirect contacts) by using trajectory data. However, there are cases where the BLE-based method is more advantageous in the contact detection capability. Since the BLE-based method relies on wireless signals, contact decisions are performed according to the constraints of the physical environment (e.g., walls, vehicles, etc.). Therefore, there is a possibility that direct contact can be detected with an accuracy that cannot be captured only by analyzing trajectory data. This is a noteworthy point compared to our method.

Reichert et al. [267] proposes a setting similar to that of our study. The similarity is that they used PCT with secret computation techniques and centralized trajectory data. Our work differs from theirs in the contact detection algorithm and the secret computation technique used. The basis of their method is a PSI

using ORAM, which has some performance problems. To compare, we extensively experiment on and evaluate the performance and accuracy of the two methods in this work.

We need to refer to the open-source project SafeTrace (`https://github.com/enigmampc/SafeTrace`) [330]. SafeTrace proposes a TEE-based method similar to our proposed method. SafeTrace focuses mainly on collecting and managing the trajectory data secretly. Currently, the development is not updated, and the part of contact detection (PCT) and notification service are not yet specified. Moreover, their work does not include any efficient PCT algorithm. We develop an algorithm for finding indirect contacts; thus, their work is orthogonal to mine. One clear difference between our proposed method and theirs is that their system collects the complete trajectory data, while our system uses information that is optimally encoded for PCT and can be applied in the stPSI efficiently. Hence, the proposed method is able to achieve optimized performance because it is focused on PCT.

A similar type of query to the proximity query is the reachability query [331, 332]. , where the query is to determine whether or not it is possible to reach a target vertex from a source vertex in a given digraph. This method has the potential to perform contact tracing by using a graph related to meeting information [332]. However, it is not suitable for answering contact judgment queries for unknown users because it can answer queries only for data that can be precomputed and are already in the database, and it cannot answer unknown data efficiently. [332] has a different feature from our proposed method because it can track contacts on a human basis, not on a location basis.

Regarding trajectory data compression, a well-studied technique is that of Douglas-Peucker, i.e., route-wise compression [333, 334, 335]. The basic idea is reproduction by estimating the route from a minimum number of points [333]. The compression is performed by approximating the route information rather than the position information. Our compression is based on the similarity of trajectory points. Therefore, it is orthogonal to these compression methods. However, this route-wise compression may not work well in contact tracing because it is not possible to tell whether or not points are really in contact just by intersecting. To detect contact, it is necessary to include some time information in the route information.

B.8 Conclusions

In this work, we proposed a trajectory-based PCT system using trusted hardware to control the spread of infectious diseases. We identified the problems of existing PCT systems, clarified the requirements for trajectory-based PCT, and presented a TEE-based architecture to achieve secure, efficient, flexible, and accurate contact tracing. The experimental results obtained on real data suggested that our proposed system can work on a realistic scale. We hope that this study will motivate different communities and help in the development of solutions to combat COVID-19 as soon as possible.